VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro indexování souborového systému
Application for a file system indexing

2009/2010 Marek Wija

Zadání diplomové práce

Téma: Aplikace pro indexování souborového systému
Diplomat: Marek Wija
Vedoucí: Ing. Radim Bača
Akademický rok: 2009/2010
Obor: 2612R025 Informatika a výpočetní technika

Zásady pro vypracování:

Dnešní aplikace umožňující rychlé vyhledávání souborů na disku jsou většinou založeny na jednoduchých parametrech jako je maska jména souboru, datum. Takto zadané vyhledávání může vrátit velké množství nerelevantních výsledků. Přibližné zadání cesty v souborovém systému by mohlo značně snížit velikost výsledku a zpřesnit vyhledávání.

V rámci této práce student navrhne a naimplementuje aplikaci, která bude sloužit k zaindexování souborů tak, aby bylo možné v nich efektivně vyhledávat i komplikovanějšími dotazy. Aplikace bude splňovat následující požadavky:

1. Aplikace bude schopna převést adresářový strom na odpovídající XML dokument, který následně uloží v XML databázi.
2. Bude sledovat změny v adresářové struktuře a tyto změny bude přenášet do XML dokumentu.
3. Umožní vyhledávat v adresářové struktuře s využitím možností jazyka XPath nebo XQuery

Postup řešení:

1. Prostudování problematiky XML databází.
2. Analýza a návrh aplikace s použitím XML databáze.
3. Implementace a otestování výsledné aplikace.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: ……

Rád bych na tomto místě poděkoval ing. Radimu Bačovi za odborné vedení.

Abstrakt

Cílem práce je návrh a implementace aplikace indexující obsah vybrané složky do XML databáze a v takto vytvořeném indexu vyhledávat pomocí dotazů jazyka XPath nebo XQuery. Výhodou vyhledávání souborů v takovém indexu je určitá flexibilita zadání podmínek, kde nejsme omezení pouze maskou souboru, ale můžeme výsledek více upřesnit zadáním přibližné cesty k souboru a tím zmenšit velikost výsledku. Ačkoliv je použitá XML databáze nejvhodnějším kandidátem z malého výběru volně dostupných nativních XML databází, neposkytuje dostatečnou rychlost pro praktické použití v tomto typu aplikace, navíc konzumuje obrovské množství paměti disku. Výkon, konzumace paměti spolu s malým množstvím dostupných nativních XML databází jen utvrzuje fakt, že v současné době je stadium vývoje XML databází teprve v počátcích, a je zaměřeno spíše na ukládání více menších XML dokumentů do databáze a následnou práci s nimi. V případě XML dokumentů s velkou hloubkou zanoření a obsahující tisíce elementů, jejich výkon velmi pokulhává.

Klíčová slova: XML, aplikace, služba, indexování, nativní XML databáze, XPath, XQuery

Abstract

The aim of the thesis is design and implementation of an application that indexes the content of selected folder to XML database and to search the following indexes queries using XPath or XQuery. The advantage of searching the files in that index is a certain flexibility in terms of the award where we are not only limited by the mask file but we are able to refine the result by entering the approximate path of the file and to reduce the size of the results by this way. Although the use of XML databases is the most appropriate candidate from a small selection of freely available native XML databases do not provide sufficient speed for practical use in this type of application, in addition consumes a huge amount of memory to disk. The performance, the memory consumption, together with a small amount of available native XML databases only confirms the fact that at present the stage of development of XML databases is only in its infancy at present and is focused on the storage of several smaller XML documents into a database and subsequent work with them. In the case of XML documents with great depth of nesting and containing thousands of elements their performance fails very much.

Keywords: XML, application, service, indexing, native XML database, XPath, XQuery

Seznam použitých symbolů a zkratek

XML – eXtensible Markup Language

W3C – World Wide Web Consortium

WWW - World Wide Web

XPath – XML Path Language

XQuery – An XML Query langure

API – Application Programming Interface

HTTP – HyperText Transfer Protokol

.NET Framework –

NXDBS – Native XML Database System

ACID - *Atomic, Consistent, Isolated, Durable*, je obecně uznávaný seznam požadavků na bezpečný transakční systém:

- Atomičnost - v rámci transakce se provedou všechny změny nebo žádná.
- Konzistence - transakce zajišťují převedení dat z jednoho konzistentního stavu do druhého. Tato podmínka nemusí platit uvnitř transakce.
- Izolace - transakce není ovlivněna souběžnými transakcemi.
- Trvanlivost - pokud je transakce potvrzená, pak jsou změny dat trvalé a to i pokud nastane havárie systému.

URI – Uniform Resource Identifier

HTML – HyperText Markup Language

CSS - Cascading Style Sheets

XHTML – eXtensible HyperText Markup Language

XSLT - eXtensible Stylesheet Language Transformations

MB – Mega Byte

GB – Giga Byte

REST - Representational State Transfer

AJAX - Asynchronous JavaScript and XML

DOM - Document Object Model

SAX - Simple API for XML

RPC – Remote Procedure Call

SOAP - Simple Object Access Protocol

OBSAH

1 Úvod

Úkolem mé práce bylo seznámit se s problematikou XML databází a vybrat si jednu, volně dostupnou XML databázi, implementující dotazovací jazyky XPath a XQuery. Protože aplikace má pracovat jako služba na pozadí, potřeboval jsem i XML databázi, která umožňuje pracovat jako služba na pozadí. Jako vhodná se ukázala XML databáze Sedna [1], vyvinutá Institutem systémového programování Ruské Akademie věd, která jako jedno z mnoha API nabízí i API určené pro programovací jazyk C#, jakožto jazyka, který jsem si vybral pro implementaci aplikace.

Naneštěstí se v případě Sedny jednalo o přídavný neoficiální modul, který Sedně umožnil spuštění jako služby. Postupem času, kdy se systém Windows XP několikrát aktualizoval, přestalo spouštění sedny jako služby pracovat správně. Sedna databáze se proto musí spouštět ručně, před spuštěním samotné služby a aplikace indexování souborů. V současné době se mi nepodařilo nalézt volně dostupnou nativní XML databázi, která by pracovala jako služba. Jedinou výjimkou byla databáze eXist [2], která pracuje na webovém serveru Jetty a ten může být spouštěn jako služba. Exist je však především zaměřena na použití ve webových aplikacích, umožňuje sice začlenit do aplikace v jazyce Java, ale vzhledem k tomu, že jako implementační jazyk byl vybrán C#, nemohl jsem ji použít.

Indexováním se zde rozumí vytvoření XML dokumentů a udržování jejich integrity s adresářovou a souborovou strukturou složek (i disků), které chceme indexovat. Tyto XML dokumenty jsou uloženy v XML databázi, která umožňuje tyto soubory rychle dotazovat pomocí dotazovacích jazyků XPath a XQuery a v případě databáze Sedny aktualizovat jazykem navrženým Patrickem Lehtiem [3] založeném na jazyku XQuery. XML dokument musí být ovšem správně strukturovaný (well-formed), v opačném případě Sedna ohlásí chybu. Snahou je tedy navrhnout strukturu XML dokumentu, který bude odrážet stromovou strukturu vybrané složky a zároveň bude správně strukturovaný.

Pro získání diskové struktury vybrané složky a vytvoření XML stromu je implementována aplikace, jejímž prostřednictvím si uživatel vybere složky, které chce indexovat. Tato aplikace se prostřednictvím API spojí s XML databází Sedna, projde indexované složky a jejich struktury zanáší do jim příslušejících XML souborů, které jsou uloženy v XML databázi Sedny. Pomocí této aplikace má uživatel také možnost zadávat dotazy pro vyhledávání v indexu.

Pro udržování integrity těchto indexů resp. XML dokumentů, je implementována aplikace běžící jako služba na pozadí, která je spojena s XML databázi Sedna pomocí API a detekuje změny v indexovaných složkách, tedy diskové operace uvnitř indexované složky. Cílem služby je promítnout tyto změny do XML souboru příslušného indexu.

Pro implementaci aplikace a služby jsem zvolil jazyk C#, který mi vyšel vstříc s již hotovou třídou pro sledování změn na disku a získávání informaci o souborech a adresářích, které jsou součástí. NET Frameworku již od verze 1.0.

V následující kapitole Nativní XML databáze práce pojednává o nativních XML databázích a výběru použité XML databáze z několika kandidátů. Ve třetí kapitole podrobně seznamuje s vybranou Sedna nativní XML databázi. V další kapitole je stručně popsán jazyk XPath a XQuery. Kapitola pátá je věnována samotnému návrhu a analýze aplikace a služby indexování souborového systému. V kapitole implementace je stručně popsáno API databáze Sedna pro jazyk C#. Předposlední sedmá kapitola se věnuje testování výsledné aplikace z hlediska aktualizace a vyhledávání v indexu. Poslední závěrečná kapitola shrnuje výsledky práce.

2 Nativní XML databáze

Nativní XML databáze jsou speciální databázové systémy přímo určené k ukládání XML dat. Také jako klasické databáze podporují transakce, zabezpečení, víceuživatelský přístup, dotazovací jazyky apod. Jediným rozdílem těchto databázových systémů je jejich vnitřní model, který je celý založen jen na XML. Při psaní této kapitoly jsem využil diplomové práce Zdeňky Albrechtové [4].

Hlavní rysy nativních XML databází se dají shrnout do třech bodů:

- Objekty k uskladnění v nativních XML databázích jsou XML data.
- XML dokument je považován za základní jednotku uložení dat.
- Nativní XML databáze ukládá XML dokumenty v „nativní“ (přirozené) formě, ale nemusí se jednat o samostatný databázový systém.

Termín „nativní XML databáze“ (v originále „native XML database“) se poprvé objevil při marketingové kampani projektu Tamino [5], nativní XML databáze od Software AG, a hned po ní se celosvětově uchytil. Bohužel doposud neexistuje žádná formální definice „nativní XML databáze“.

2.1 XML dokumenty

Do XML dokumentů lze ukládat data různého původu. V praxi se data ukládaná do XML rozdělují na dokumentově orientovaná (dokument-centric) a datově orientovaná (data-centric). V závislosti na uložených datech se pak vyskytují i dva druhy XML dokumentů. Úplnou specifikaci XML dokumentu popisuje W3C [6].

2.1.1 Datově orientované XML dokumenty

Datově orientované XML dokumenty slouží především pro přenos dat. Většinou jsou určeny pro využití v jiných programech, zpracování lidmi se příliš nepředpokládá. Příklady takovýchto dokumentů: objednávky, faktury, jízdní řády, vědecká data (např. záznamy měření), …

Charakteristickým znakem těchto dokumentů je pravidelná struktura, co nejmenší logické jednotky dat v elementech a atributech a malý nebo žádný výskyt smíšeného obsahu.

2.1.2 Dokumentově orientované XML dokumenty

Dokumentově orientované XML dokumenty jsou dokumenty obvykle využívané pro lidskou potřebu. Velmi dobrým příkladem jsou knihy.

Vyznačují se méně pravidelnou, často až nepravidelnou, strukturou, hrubým členěním dat a velkým výskytem smíšeného obsahu. Nejčastěji jsou tyto dokumenty psány ručně přímo v XML nebo v jiných formátech, jako např. RTF, PDF, …, které jsou následně do XML převedeny.

2.1.3 Použitý typ XML dokumentu

XML dokumenty ukládané do nativních XML databází mohou být libovolné struktury, v praxi se však do těchto systémů ukládají častěji dokumentově orientované XML dokumenty, protože s využitím dotazovacích jazyků dostupných právě v těchto databázových systémech je možné rychle a jednoduše získat odpověď na téměř jakoukoli otázku. Dalším důvodem pro použití těchto systémů je zachovávání struktury, procesních instrukcí, sekcí CDATA, komentářů, což databáze založené na XML (XML-enabled Database[1]) neumožňují.

2.2 Popis vlastností vybraných XML databází

Při hledání vhodné XML databáze jsem vycházel z bakalářské práce Radoslava Činčaly [7], který testoval čtyři XML databáze z hlediska výkonu. Jedná se o XML databáze MonetDB [8], X-Hive [9], Sedna [10] a Timber [11]. Také jsem sám vyzkoušel dvě volně dostupné XML databáze BaseX [12], eXist [2], jejichž vlastnosti popíši v následujících dvou kapitolách.

2.3 BaseX

BaseX je open source, napsaný v Javě a volně dostupný ke stažení. Je vyvíjen Skupinou zabývající se vývojem databází a informačních systémů na univerzitě v Konstanz v Německu.

[1] Rozdíl mezi databází nativní a založené na XML je v ukládání XML dokumentu. XML-enabled používá k uložení XML dokumentu specifické schéma, které musí být aplikováno na XML dokument v době jeho návrhu (vzniku).

2.3.1 Vlastnosti jádra databáze:

- Vyspělá optimalizace dotazu k aplikování indexových struktur kdykoli je to vhodné
- Import HTML, jestliže je v cestě třídy zahrnut TagSoup [13] – prochází kód HTML a opravuje chybně zanořené HTML značky
- Kompaktní, vysoce výkonná databáze umožňující indexovat text, atributy, cesty a full-textové indexování.
- Klient/server architektura, podpora ACID transakcí, správu uživatelů, logování
- Vysoce interaktivní vizualizaci, podporující velmi velké XML dokumenty
- XQuery editor se zvýrazněním syntaxe a hlášením chyb

2.3.2 XML standardy:

- Podpora XSLT/XQuery serializace [14]
- Větší shoda s XQuery Update/Full Text Doporučeními [15] [16]
- Účinná podpora W3C XPath/XQuery doporučení spolu s Full text a Update rozšířeními
- Od verze 6.0 implementuje kompletní W3C XQuery Update Facility

2.3.3 API:

- Poskytuje vazby na několik programovacích jazyků, to umožňuje vytváření klientů v různých programovacích jazycích a připojení k běžícímu BaseX serveru. Pomocí klienta můžeme pak spouštět všechny databázové příkazy na serveru a přijímat výsledky. K dispozici jsou následující API pro:
  - Objektově orientované jazyky - Java, C#, VB, Boo
  - Skriptovací jazyky – PHP, Python, Perl, Ruby, Rebol
  - Funkcionální jazyky – Lisp, Haskell

2.3.4 Síťové protokoly:

- Nové REST API, založené na JAX-RX[2] [17] rozhraní pro komunikaci BaseX pomocí http protokolu, více o použití rozhraní REST lze nalézt na [18].

[2] JAX-RX reprezentuje tenkou vrstvu založenou na REST rozhraní určenou pro jednotný přístup k databázím a zdrojům.

2.4 eXist

Exist je volně dostupný, databázový řídicí systém postaven na technologii XML a implementován v jazyce Java. Celý systém běží uvnitř webové aplikace (v podstatě je webovou aplikací) spuštěné na před-konfigurovaném serveru Jetty [19]. Všechna rozhraní jsou poskytována jako servlety.

Administraci i dotazování v eXist můžeme provádět bud pomocí webového rozhraní nebo pomocí administračního klienta napsaného v Javě. Jelikož tato databáze běží na webovém serveru Jetty a ten umožňuje být spuštěn jako služba, můžeme říci, že tato databáze umožňuje běh na pozadí, tedy jako služba.

Exist poskytuje výkonné prostředí pro vývoj webových aplikací založených na XQuery a příbuzných standardů. Celá webová aplikace může být napsána v XQuery, užitím XSLT, XHTML [20], CSS [21] a Javascriptu [22](AJAX [23]). XQuery stránky mohou být vykonávány ze souborového systému nebo uloženy v databázi.

Podporuje mnoho (webových) technologických standardů:

- XQuery, XPath, XSLT
- HTTP rozhraní REST , WebDAV [24], SOAP [25], XMLRPC [26], Atom protokol [27]
- Rozšíření XQuery pro aktualizaci srovnatelné s XQuery Update Facility 1.0 [15].

2.4.1 Vlastnosti jádra databáze:

- **Ukládání dat** - Ukládání nativních XML dat je založeno na B+-stromu a stránkovaných souborech. Uzly dokumentu jsou ukládány v perzistentním DOM [28].
- **Kolekce** - Dokumenty jsou spravovány v hierarchických kolekcích, podobně jako ukládání souborů v souborovém systému. Kolekce nejsou vázány na předefinované schéma nebo typ dokumentu.
- **Indexování** - Založeno na číselném indexovacím schématu, které podporuje rychlou identifikaci ve struktuře vztahů mezi uzly, jako jsou rodič/dítě, předek/potomek nebo předchozí-/další-sourozenec.
- **Vytváření indexu** - Index se vytváří implicitně automaticky. Používá strukturální indexy pro elementy a atributy uzlu. Fulltextový index pro text a hodnoty atributů, řadu

indexů pro typové hodnoty. Fulltextové indexování může být vypnuto/zapnuto pro vybrané části dokumentu. Strukturální indexy jsou udržovány automaticky.

- **Dotazovací engine** - eXist má svůj vlastní optimalizovaný dotazovací stroj, účinné na indexech založené zpracování dotazu. V rozporu s tradiční realizací, se eXist snaží předcházet průchody stromem metodou shora dolů nebo ze spodu nahoru. Místo toho spoléhá na rychlé algoritmy spojování cesty k výpočtu vztahů uzlu. Spojování cesty může být aplikováno na celou sekvenci uzlu v jednom kroku.
- **Aktualizace** - Aktualizace pomocí XUpdate a XQuery a update rozšířením.
- **Autorizační mechanismus** - Podobně jako v Linuxu přístupová práva pro uživatelé/skupiny v kolekcích a na úrovni dokumentu.
- **Bezpečnost** - Jazyk eXtensible Access Control Markup Language (XACML) [29] pro XQuery řízení přístupu.
- **Nasazení** - Může být nasazen jako samostatný databázový server, jako zabudovaná Java knihovna, nebo jako část webové aplikace.
- **Záloha/Obnova** - Funkce zálohování/obnovy je poskytována prostřednictvím Java admin klienta nebo skriptu pro Ant [30]. Záloha obsahuje zdroje jako čitelné XML. Umožňuje plně obnovit databázi včetně uživatelskými/skupinovými přístupy.
- **Transakce/zotavení** - Databáze podporuje plné zotavení po pádu založené na deníku, ve kterém jsou všechny transakce zaznamenávány. V případě pádu databáze, jsou všechny závazné transakce znovuzpracovány zatímco všechny nekompletní transakce jsou vráceny zpět. Nicméně, podpora transakcí je limitována funkcionalitou potřebnou pro obnovu při pádu. To znamená, že transakce nejsou viditelné nebo použitelné z aplikačního kódu. Všechny transakce jsou vytvářeny automaticky různými API.

2.4.2 XML standardy:

- **XPath/XQuery** - XQuery 1.0. Dotazy mohou být vztaženy na jakoukoli kombinaci kolekcí nebo dokumentů.
- **XQuery moduly** - XPath/XQuery standardní knihovna funkcí a mnoho dalších přídavných modulů pro manipulaci s obsahem databáze, dynamické vyhodnocování XQuery výrazů, XSL transformace, funkce související s HTTP. Uživatelské moduly mohou být poskytnuty v XQuery nebo v Javě.
- **XInclude** [31] - Podporováno mimo některé vlastnosti.

- **XPointer** [32] - Podporováno částečně.
- **XSL/XSLT** - Podporováno prostřednictvím rozhraní serveru nebo rozšiřujícími funkcemi XQuery.
- **XUpdate** - Ano a také update syntaxe rozšiřující XQuery.

2.4.3 API:

- **XML:DB API** - Poskytuje ovladače pro přístup k vzdálené databázi nebo pro vestavěné použití. Obsahuje dodatečné služby pro správu uživatelů, správu kolekcí.
- **DOM** - Prostřednictvím XML:DB API. Poskytnut přímý (čtení) přístup k zabudovaným databázím k perzistentnímu DOM.
- **SAX** - Prostřednictvím XML:DB API.

2.4.4 Síťové protokoly:

- **HTTP/REST** - eXist nabízí REST-API pro jednoduchý přístup pomocí HTTP protokolu. Obecně nejrychlejší cesta přístupu k databázi.
- **XML-RPC** - Preferovaný protokol užívaný XML:DB API ovladači. Poskytuje plný přístup ke všem funkcím databáze.
- **SOAP** - Podporován v servlet módu. Založen na Apache Axis [33]
- **WebDAV** - Kompletní podpora pro všechny hlavní operace WebDAV

2.4.5 Limity:

- **Maximální počet dokumentů** - Maximální počet uložených dokumentů v databázi je 2^{31}.
- **Maximální velikost dokumentu** - Teoreticky libovolně veliký, v praxi, některé vlastnosti jako počet dětí uzlu jsou limitovány čtyř bajtovým celým číslem. Také omezení operačního systému jako jsou maximální velikost souboru v souborovém systému a další.

2.5 Použitá XML databáze a API

Pro ukládání XML dokumentů jsem si vybral volně dostupnou nekomerční XML nativní databázi Sedna, která jediná z volně dostupných umožňuje běh na pozadí. Také z hlediska výkonu a paměťových nároků tato databáze vyšla nejlépe. Tímto se odkazuji na bakalářskou práci Radoslava Činčaly [7], který testoval čtyři XML databáze z hlediska výkonu. Nejrychlejší z testovaných byla databáze MonetDB [8], ale z hlediska paměťových nároků, byla pro mou aplikaci nepoužitelná. Dalšími testovanými byly X-Hive [9] a Sedna [10], které byly z hlediska výkonu na stejné úrovni, ale X-Hive je vázaná licencí. Poslední byla databáze Timber [11], která činila problémy, takže její použití nepřipadalo v úvahu. Mnou vyzkoušená XML databáze eXist, nenabízí programové rozhraní pro jazyk C# a navíc je spíše zaměřena na vývoj webových aplikací psaných v jazyce Java. Další XML databáze BaseX neumožňuje běh na pozadí.

3 Sedna XML databáze

Volně dostupná NXDBS [10], který poskytuje množství databázových služeb – trvalé úložiště dat, ACID transakce, bezpečnost, indexy, zálohy

3.1 Základní vlastnosti

- Spadající pod Apache Licence 2.0, volně dostupná open source
- Nativní XML databázový systém implementován v jazyce C/C++
- Podporuje W3C XQuery dotazovací jazyk splňující W3C XQuery Test Suite
- Full-text vyhledávání integrováno do XQuery, založené na dtSearch
- Podpora deklarativního[3] aktualizačního jazyka na úrovni uzlu
- Podpora ACID transakcí
- Podpora XML triggerů (spouštěčů)
- Účinné přírůstkové zálohování
- Indexy (založené na B-stromu)
- Podpora UTF-8 (Unicode)
- SQL propojení s XQuery – umožňuje XQuery přístup k relačním databázím užitím SQL
- XQuery externí funkce implementované v C
- Bezpečnost (uživatelé, role, práva)
- Modul pro integraci s Apache HTTP serverem
- Aplikační programové rozhraní pro Javu, C, PHP, Python, Ruby, C# a další

3.2 Architektura XML databáze

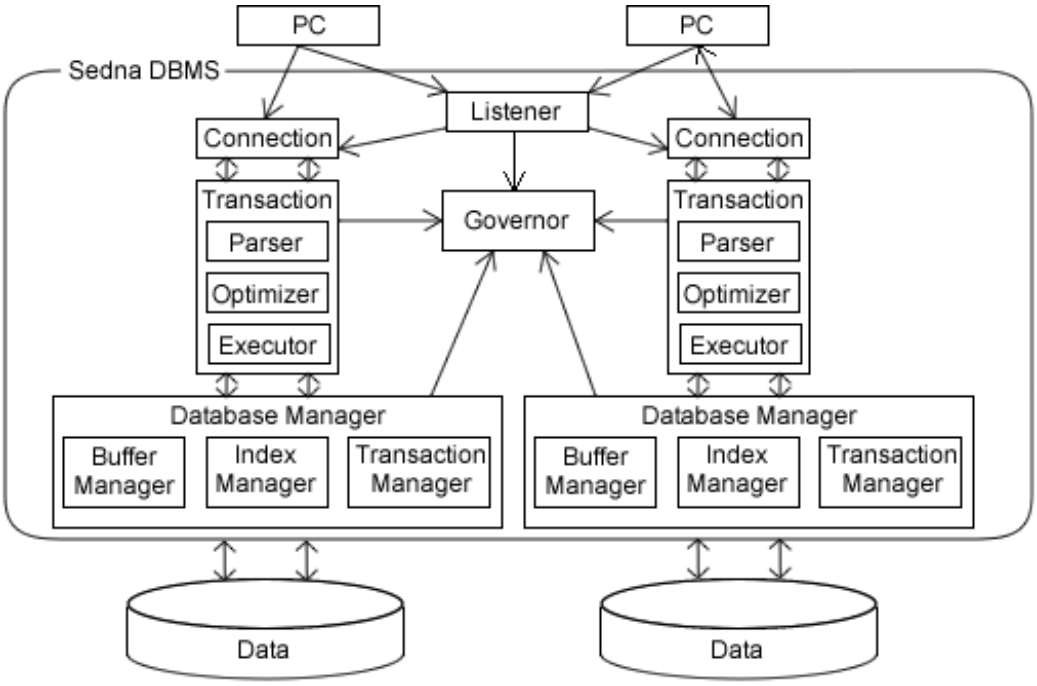
[1]

Na obrázku 1 je zobrazena architektura XML databáze Sedna. Skládá se z následujících komponent:

- **governor** slouží jako řídící centrum systému, kde se všechny ostatní komponenty registrují. Kontroluje a řídí všechny běžící transakce a databáze v systému.

[3] Deklarativní programování je založeno na myšlence programování aplikací pomocí definic, co se má udělat a ne jak se to má udělat. Opakem tohoto principu je imperativní programování popisující jednotlivé úkony pomocí algoritmů. Zjednodušeně to lze popsat tak, že imperativní programy obsahují algoritmy, kterými se dosáhne chtěný cíl, zatímco deklarativní jazyky specifikují cíl a algoritmizace je ponechána programu (interpretu) daného jazyka. [38]

- **listener** vytváří instanci komponenty připojení (Connection component, dále CC) pro každého klienta a založí přímé spojení mezi klientem a CC. CC zapouzdřuje klientské sezení (session) a pro každého klienta, který začíná transakci, vytváří instanci komponenty Transaction.
- **transaction** vykonává dotazy a skládá se z:
  - **parser** překládá dotaz do své logické reprezentace, tzv. stromu operací.
  - **optimizer** přebírá logickou reprezentaci dotazu a produkuje optimalizovaný spouštěcí plán, který je stromem nízko úrovňových operací, prováděných nad fyzickou strukturou dat.
  - **exekutor** interpretuje výše zmíněný spouštěcí plán
- **database manager** každá instance zapouzdřuje pro jednu databázi služby jako správce indexů (index manager), který udržuje indexy založené na databázi, správce vyrovnávací paměti (buffer manager), který je zodpovědný za interakce mezi diskem a operační paměti a správce transakcí (transaction manager), která řídí souběžné vykonávání transakcí.



Obrázek 1: Architektura Sedna NXDBS

3.3 Řízení souběžnosti transakcí

Sedna podporuje víceuživatelský souběžný přístup k datům. Používá protokol zamčení, k řešení nejrůznějších synchronizačních problémů, zamykání se děje na úrovni dokumentů a uzlů v dokumentu. K zajištění sériového vykonávání transakcí používá známý dvou fázový zamykající protokol. Jelikož v mnoha případech není zapotřebí zamykat celý dokument, Sedna umožňuje zamykat také jen uzly a celé podstromy dokumentu.

3.4 Datové soubory

Po vytvoření a používání jedné nové databáze vznikají tyto soubory:

- <jmeno_db>.sedata – sloužící k trvalému uložení XML dat
- <jmeno_db>.setmp – ukládají se zde dočasné mezivýsledky vznikající během spuštění dotazu
- <jmeno_db>.*llog – soubory k ukládání logů Sedny

Při vytváření databáze lze zadat i její maximální velikost, implicitně je nekonečná. Velkou nevýhodou zjištěnou během vyvíjení aplikace je, že datový soubor se pouze zvětšuje. Při tomto zjištění jsem se opět obrátil na tzv. Sedna mailing list, který se zabývá řešením problémů Sedny. Dočkal jsem se odpovědi: „Smazání dokumentu zabere nové bloky, které dokument zabírá (díky mechanizmu verzovaní), v tuto chvíli není nic, co s tím můžete dělat. Neexistuje žádný způsob, jak zmenšit databázový soubor. Jediné dobré je, že všechno uvolněné místo obsazené smazaným dokumentem bude znovu použito při nahrání nového dokumentu (stejné nebo menší velikosti), databázový soubor by se neměl zvětšit.“

Odstraněním dat nebo i celých dokumentů z databáze, nedojde ke zmenšení datového souboru. Navíc se datový soubor zvětšuje velice rychle a závisí na hloubce stromu dokumentu. Pro příklad po zaindexování přibližně 31000 souborů a složek měl databázový soubor velikost 7 GB. Vyhledávání a změny v takto velkém souboru, pokud zrovna potřebná část databáze není umístěna v operační paměti, již značně trvá a záleží na výkonu pevného disku, jak rychle je schopen dodávat další potřebné části do operační paměti.

Na rozdíl od například MonetDB [7], Sedna nenahrává celou databázi do paměti. Každá databáze má vlastní konfigurační soubor, ve kterém lze mimo jiné, specifikovat velikost paměti, do které bude umístěna aktuálně používaná část databázového souboru. Tato nastavení, je možno provést ihned při vytváření nové databáze. Čím více paměti přiřadíme, tím rychleji bude Sedna pracovat, záleží ovšem na tom, kolik si ji můžeme dovolit, jakou velikostí disponuje naše PC. Platí ovšem také, že čím více paměti přiřadíme databázi, tím déle bude trvat její první načtení z disku.

Pro použití v aplikaci jsem nastavil velikost přiřazené paměti na 600 MB.

3.5 Dotazovací jazyk

Pro dotazování dat v databázi je použit jazyk XQuery, který je popsán v kapitole 4.

3.6 Aktualizační jazyk

Sedna XML databáze používá pro aktualizaci uložených XML dokumentů jazyk založený na XQuery speciálně upravený Patrikem Lehtim. Tento jazyk je využíván službou při indexování souborů pro aktualizaci indexu. Podle zachycené události v indexované složce jsou vytvářeny následující konstrukce.

INSERT

Vložení nového uzlu do indexu. Použití příkazu insert nastává, vznikne-li v indexované složce nová složka nebo nový soubor. Syntaxe příkazu insert je následující:

UPDATE
insert *výraz1* into *výraz2*

Provede vložení sekvence uzlů dané *výrazem1* do všech uzlů dané výsledkem *výrazu2*.

Příklad:

```
UPDATE
insert <novýSoubor path='c:\cesta\novýSoubor' /> into doc('dokument','kolekce')/index/cesta
```

Provede vložení nového elementu novýSoubor s atributem path do uzlu cesta do dokumentu dokument v kolekci kolekce.

DELETE

Odstranění uzlu z indexu. Použití příkazu delete nastává, dojde-li v indexované složce k odstranění složky nebo souboru. Syntaxe příkazu delete je následující:

UPDATE
delete *výraz*

Odstraní uzly dané výsledkem *výrazu.*

Příklad:

```
UPDATE
delete doc('dokument','kolekce')/index/cesta/soubor
```

Odstraní z indexu uzel soubor nacházející se v uzlu cesta.

REPLACE

Nahrazení stávajícího uzlu uzlem novým. Použití příkazu replace nastává, při přejmenování souboru či složky. Syntaxe je následující:

UPDATE
replace *$p výraz1*
with *výraz2($p)*

Všechny uzly dané zpracováním *výrazu1* jsou nahrazeny uzly dané *výrazem2*. Proměnná *$p* je svázána s uzly danými *výrazem1*.

Příklad:

```
UPDATE
replace $p in doc('dokument','kolekce')/index/cesta/staréJménoSouboru
with
<novéJménoSouboru path='c:\cesta\novéJménoSouboru' />
```

Provede nahrazení elementu staréJménoSouboru novým elementem novéJménoSouboru, který obsahuje i novou aktuální absolutní cestu k souboru na disku.

4 XPath a XQuery

Jelikož je aktualizační (update) jazyk Sedna XML databáze založen na XQuery, který používá datový model XPath uvedu jemný úvod do těchto jazyků určených k dotazování XML dokumentů.

4.1 XPath

- XPath je jazyk primárně určený k adresování částí XML dokumentu
- byl navržen pro použití s XSLT transformací a XPointer
- poskytuje prostředky pro manipulaci s řetězci, čísly a booleovskými hodnotami
- používá kompaktní ne-XML syntaxi k usnadnění použití XPath uvnitř URI a hodnot XML atributů
- pracuje na abstraktní, logické struktuře XML dokumentu
- modeluje XML dokument jako strom uzlů
- více o XPath lze nalézt v jeho specifikaci [34]

4.1.1 Výraz

Výraz (expression) je primární syntaktická konstrukce XPath. Vyhodnocení výrazu přinese objekt, který má jeden z těchto čtyř základních typů:

- Množinu uzlů (node-set), (netříděná kolekce uzlů bez duplicit)
- Booleovská hodnota (true, false)
- Číslo (včetně desetinné čárky)
- Řetězec (sekvence znaků)

K vyhodnocení výrazů slouží knihovna funkcí, která obsahuje funkce pro práci s řetězci, čísly a booleovskými hodnotami.

4.1.2 Datový model

XPath modeluje XML dokument jako strom. Tento strom může obsahovat 7 typů uzlů:

- Kořenový uzel (root)
- Element
- Text
- Atribut

- Prostory jmen (namespace)
- Uzel zpracovávající instrukce
- Komentáře

Pro každý typ uzlu, existuje cesta určující jeho hodnotu řetězce (dále obsah). Pro některé typy uzlu je obsah částí uzlu, pro jiné typy uzlu je obsah vypočítán z obsahu následujících uzlů (potomků).

4.1.3 Syntaxe

Výběr uzlů

XPath používá k výběru uzlů v XML dokumentu výraz v podobě cesty. Hledaný uzel můžeme například nalézt aplikací výrazů:

- Jméno uzlu – vybere všechny potomky tohoto uzlu
- / - pokud začneme cestu znakem lomeno, začínáme vyhledávat od kořenového uzlu
- // - pokud použijeme ve výrazu cesty dvojité lomeno, vybereme všechny uzly odpovídající výrazu za lomítkem, bez ohledu na to, v jaké hloubce od aktuálního uzlu se nachází, jsou to jeho všichni potomci
- . – vybere aktuální uzel
- .. – vybere rodiče aktuálního uzlu
- @ - vybere atributy

Jestli-že cesta začíná lomítkem, jedná se o absolutní cestu od kořene k uzlu.

Predikáty

Ve výrazech lze používat predikáty k nalezení specifického uzlu nebo uzlu obsahující specifické hodnoty. Predikáty jsou vždy umístěny v hranatých závorkách. Nejlépe je použití predikátů ilustrovat na příkladech:

- /kořen/uzel[1] – vrátí první výskyt uzlu uzel, který je obsažen v uzlu kořen
- /kořen/uzel[last()] – vrací poslední výskyt uzlu uzel, který je obsažen v uzlu kořen
- /kořen/uzel[last() - 1] – vrací předposlední výskyt uzlu uzel, který je obsažen v uzlu kořen
- /kořen/uzel[position() < 3] – vrací první dva výskyty uzlu uzel, který je obsažen v uzlu kořen
- /kořen//uzel[@jazyk] – vrací všechny uzly uzel obsahující atribut jazyk, obsažené v uzlu kořen, nehledě na úroveň zanoření
- /kořen//uzel[@jazyk = ”čeština”] – vrací všechny uzly uzel obsahující atribut jazyk, který je roven čeština, obsažené v uzlu kořen, nehledě na úroveň zanoření
- Další příklady použití predikátů lze najít na [34]

Vyhledávání neznámých uzlů

Pro vyhledávání neznámých XML elementů můžeme použít zástupných znaků, které jsou:

- * - jakýkoli element
- @* - jakýkoli atribut
- node () – jakýkoli element jakého-li druhu

příklady:

- /kořen/* - vrací všechny přímé potomky uzlu kořen, tzv. děti.
- //* - vrací všechny elementy dokumentu
- //kořen/[@*] – vybere všechny elementy s názvem kořen obsahující jakýkoli atribut

Výběr více cest najednou

Užitím logického operátor nebo „|“ ve výrazu můžeme vybrat více cest zároveň, příklad:

/kořen/uzel[2] | /kořen/uzel[10] – vybere 2. a 10. výskyt uzlu uzel obsažený v uzlu kořen

Osy XPath

Ve výrazech můžeme používat osy XPath. Osy XPath definují množiny uzlů vzhledem k aktuálnímu uzlu. V tabulce 1 jsou uvedeny názvy os a popsány množiny uzlů, jež obsahují.

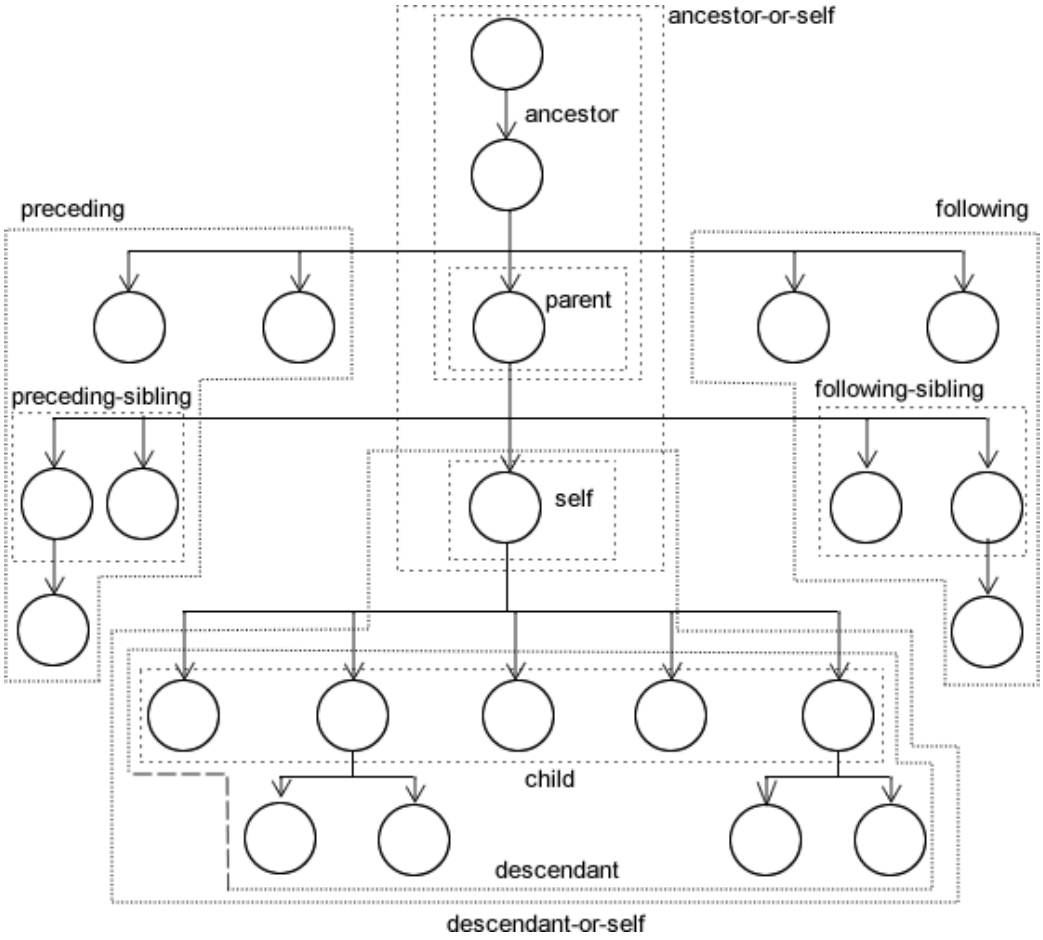
Název osy	**Výsledná množina**
ancestor	Vybere všechny předky (rodiče, prarodiče, atd.) aktuálního uzlu
ancestor-or-self	Vybere všechny předky (rodiče, prarodiče, atd.) aktuálního uzlu a aktuální uzel
attribute	Vybere všechny atributy aktuálního uzlu
child	Vybere všechny přímé následníky (děti) aktuálního uzlu
descendant	Vybere všechny následníky (potomky, potomci potomků, atd.) aktuálního uzlu
descendant-or-self	Vybere všechny následníky (potomky, potomci potomků, atd.) aktuálního uzlu a aktuální uzel (sebe sama)
following	Vybere vše v dokumentu následující po uzavírací značce aktuálního uzlu
following-sibling	Vybere všechny sourozence po aktuálním uzlu
parent	Vybere rodiče aktuálního uzlu
preceding	Vybere vše v dokumentu vyskytující se pře aktuálním uzlem
preceding-sibling	Vybere všechny sourozence vyskytující se před aktuálním uzlem
self	Vybere aktuální uzel (sebe sama)

Tabulka 1: Osy XPath

Použití XPath os:

- jméno osy::identifikátor uzlu[predikát]
- příklad: //uzel//atribute::node()[2] – vybere druhý atribut v pořadí všech elementů uzel v dokumentu

Vztahy mezi XPath osami znázorňuje obrázek 2.



Obrázek 2: Osy XPath

XPath operátory

V následujícím seznamu uvádím operátory použitelné ve výrazech XPath:

- | - sjednocení, používá se pro použití více cest pro výběr uzlů
- +, -, *, div – sčítání, odčítání, násobení, dělení
- =, !=, <, <=, >, >= - rovná se, nerovná se, menší, menší nebo rovno, větší, větší nebo rovno
- and, or – logický součin, logický součet
- mod – modulo, zbytek po celočíselném dělení

4.1.4 Funkce

XPath obsahuje mnoho funkcí podrobně popsaných v [34], uvedu příklad některých z nich, které pracují:

- s čísly
  - abs(číslo) – absolutní hodnota
- s řetězci
  - contains(“řetězec1”,”řetězec2”) – vrací true obsahuje-li řetězec1 řetězec2
- s booleovskými hodnotami
  - not(arg) - argument je nejdříve převeden na booleovskou hodnotu a poté je vrácena jeho negace
- datem a časem
  - dateTime(datum, čas) – konvertuje argumenty na datum a čas
- a další funkce

4.2 XQuery

Je jazyk určený k dotazování XML dat, nikoliv pouze XML dokumentu, ale čehokoli co se může jevit jako XML, včetně databází. Tento jazyk je postaven na XPath výrazech, sdílí stejný datový model, funkce a operátory XPath.

XQuery může být použit:

- K extrahování informace u webových služeb
- Vytváření shrnujících zpráv
- Transformace XML dat do XHTML
- Vyhledávání důležitých informací ve webových dokumentech

XQuery je kompatibilní s několika W3C standardy (XML, prostory jmen, XSLT, XPath, XML Schéma) a 23. ledna 2007 se stává normou doporučovanou W3C [35].

4.2.1 Funkce

XQuery používá pro získávání uzlů z XML dokumentů funkce. Funkce *doc()* je používána pro otevření XML dokumentu. Příklad:

- doc(“soubor.xml”) – otevře soubor soubor.xml

4.2.2 Cesta

Podobně jako v XPath se v XQuery používají pro vyhledávání elementu v dokumentu cesty. Příklad:

doc(“soubor.xml”)/kořen/uzel/potomekUzlu – vybere všechny elementy potomekUzlu, které jsou potomkem elementu uzel, které jsou potomky elementu kořen

4.2.3 Predikáty

K upřesnění cesty podobně jako v XPath můžeme použít predikátů. Příklad

doc(“soubor.xml”)/kořen/uzel[číslo > 10] – vybere pouze ty elementy s názvem uzel, které obsahují element číslo, který uvnitř obsahuje číslo větší než 10

4.2.4 FLWOR – For, Let, Where, Order by, Return

Následující XPath výraz vybere všechny elementy osoba elementu lidé, které jsou starší 20 let:

```
doc(“lide.xml”)/lidé/osoba[věk > 20]/příjmení
```

Následující konstrukce XQuery provede totožnou věc:

```
for $x in doc(“lide.xml”)/lidé/osoba
where $x/věk > 20
return $x/příjmení
```

Klauzule ***for*** provede výběr všech elementů osoba elementu lidé do proměnné ***$x***

where **-** vybere pouze ty elementy osob, které obsahují element věk, který obsahuje hodnotu větší než 20

order by - definuje seřazení výsledku, v tomto případě třídíme podle elementu příjmení

return - specifikuje, co by mělo být vráceno, v našem případě vrací element příjmení

4.2.5 Syntaxe

Základní pravidla syntaxe XQuery

- XQuery je citlivé na velikost písma
- Názvy elementů, atributů, a proměnných musí být XML validní
- Řetězec může být uzavřen v jednoduchých nebo dvojitých uvozovkách
- Proměnná je definována znakem dolar ($) následována jménem
- Komentáře jsou odděleny „(:“ a „:)“, příklad: (: text komentáře :)

Podmínky „If-Then-Else“

Příklad:

```
for $x in doc("knihy.xml")/knihovna/kniha
return if ($x/@kategorie="dětské")
then <dítě>{data($x/název)}</dítě>
else <dospělý>{data($x/název)}</dospělý>
```

Jestli-že kniha, patří do kategorie dětských, budou výsledné názvy knih uvnitř párové značky <dítě>, jinak budou uvnitř značky <dospělý>. U této konstrukce jsou použity složené závorky, které umožňují výsledek kombinovat s dalšími nově přidanými značkami. Můžeme si tak například vytvořit HTML seznam pomocí značek <ul> a <li>. Uvnitř složených závorek je použita funkce data, která vrací obsah elementu.
Nutno podotknout, že při konstrukci jsou kulaté závorky za klíčovým slovem if vyjadřující podmínku povinné. Klíčové slovo else je povinné. Pokud se ovšem pro alternativní možnost nemá vracet nic, uvedeme za slovem else prázdné kulaté závorky, tj. else().

Porovnávání

V XQuery máme dvě možnosti porovnávání

- Všeobecné srovnání: =, !=, >, >=, <, <=
- Srovnání hodnotou: eq, ne, gt, ge, lt, le

Rozdíl porovnání ukážu na příkladu:

```
$knihovna//kniha/@q > 10
```

Podmínka vrací pravdu (true) pro atributy q, které obsahují číslo větší než 10

```
$bookstore//book/@q gt 10
```

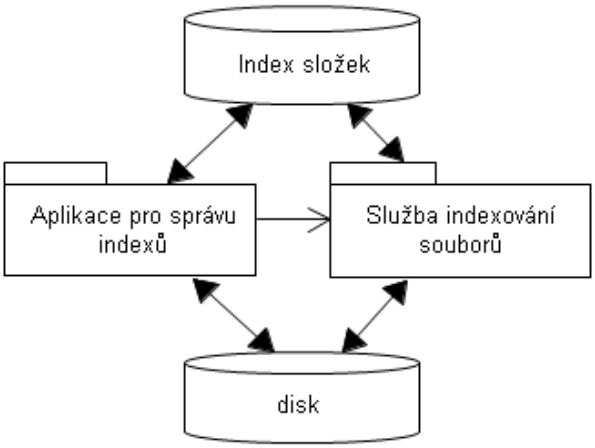
Podmínka vrací pravdu (true), pokud existuje pouze jeden atribut daný vyhodnocením výrazu a jeho hodnota je větší než 10. Pokud je vrácen více než jeden atribut q, nastane chyba.

5 Analýza a návrh aplikace

Cílem této práce je vytvořit službu, která na pozadí indexuje vybrané složky. Ke správě indexovaných složek je však potřeba vyvinout i Aplikaci pro správu indexů umožňující:

- Vybrat a přidat složku k indexování
- Odebrat indexovanou složku z indexu
- Spouštět a zastavovat indexování na vybrané složce
- Komunikovat se Službou indexování souborů

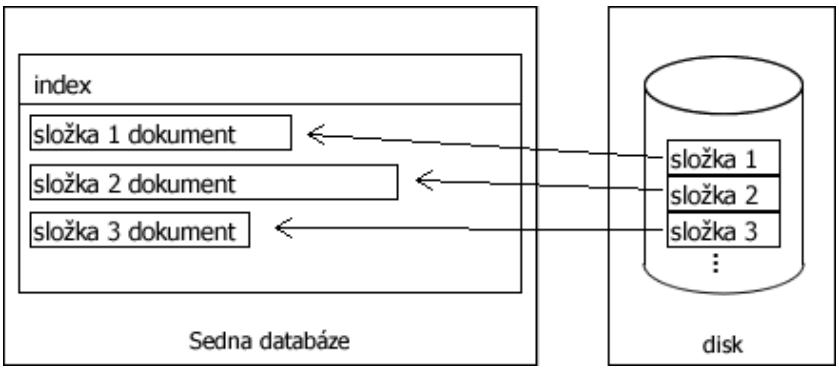
Popis komunikace Aplikace pro správu indexů a Služby indexování souborů a proudu dat (zobrazují plné šipky) je ilustrován na obrázku 3.



Obrázek 3: Komunikace

5.1 Způsob indexování složky do databáze.

Sedna umožňuje ukládání dokumentů s podobnou strukturou do kolekcí. V této kolekci následně vyhledávat v rámci všech dokumentů uložených v této kolekci. Této možnosti jsem využil a tak všechny dokumenty indexovaných složek jsou uloženy v jedné kolekci. V dalším textu budu používat pro kolekci všech indexovaným složek pojem index. Pro každou nově přidanou složku určenou k indexování, vytvoří Aplikace pro správu indexů nový, dočasný XML dokument obsahující stromovou strukturu jejího obsahu a ten nahraje do indexu. Budeme-li chtít později složku z indexu odstranit, jednoduše smažeme jí příslušející dokument z indexu. Myšlenku znázorňuje obrázek 4.



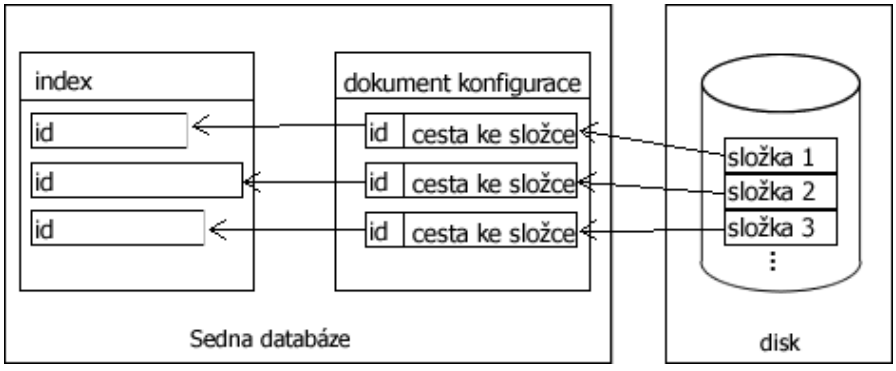
Obrázek 4: Správa indexů

5.2 Správa indexů

Pro potřebu spravování indexů je jasné, že pouze uložení dokumentů do indexu, tak jak je naznačeno na obrázku 4, nebude stačit. Je nezbytné někde zaznamenat, do kterého dokumentu z indexu se bude složka indexovat, tedy asociace znázorněné na obrázku 4 šipkami.

Pro ukládání těchto asociací je v databázi vytvořen dokument „konfigurace“, jehož struktura bude popsána dále. Propojení indexů se složkami znázorňuje obrázek 5.

Dokument konfigurace obsahuje informace o umístění složky na disku, jedinečné id, a zda je aktivní indexování. Id slouží jako název dokumentu v kolekci.



Obrázek 5: Správa indexů

5.2.1 Konfigurace indexování

Aplikace pro správu indexů při přidávání nové složky určené k indexování, vytvoří v dokumentu konfigurace nový uzel <index>, který obsahuje výše zmíněné informace popisující složku. Před vložením nového uzlu vytvoří nové id, které bude přiřazeno novému dokumentu složky. Po vložení informace do dokumentu konfigurace, spustí první zaindexování právě přidané složky. Prvotnímu zaindexování a struktuře indexu se věnuje následující podkapitola 5.3. Odstranění indexování složky odstraní uzel s požadovaným id v dokumentu konfigurace a odstraní dokument se jménem id z indexu.

Struktura dokumentu konfigurace (příklad):

```
<?xml version="1.0" standalone="yes"?>
<indexy>
    <index>
        <id>2</id>
        <cesta>D:\Dokumenty\2009</cesta>
        <aktivni>0</aktivni>
    </index>
</indexy>
```

5.3 Prvotní indexování, struktura indexu

5.3.1 Struktura indexu

Tvorba XML dokumentu odpovídajícího struktuře obsahu indexované složky se řídí předpisem:

- indexovaná složka může obsahovat pouze složky a soubory
- pro každou složku je v dokumentu vytvořen element nesoucí její jméno
- pro soubor je v dokumentu vytvořen element se shodným názvem souboru a navíc obsahující atribut path, který obsahuje úplnou cestu k souboru na disku
- elementy složek mohou obsahovat vnořené elementy složek (uzly) a souborů (listy), tvoří strom odpovídající struktuře složky na disku
- elementy souborů nemohou obsahovat vnořené elementy, tvoří listy stromu

Index, který je zde XML dokument musí být správně strukturován. Musí splňovat specifikaci W3C. A tady nastaly problémy.

Správné pojmenování elementů:

- Jméno může obsahovat písmena, čísla, jiné znaky
- Jméno nesmí začínat číslicí nebo interpunkčními znaky
- Jméno nesmí začínat písmeny xml (Xml,XML,atd.)
- Jméno nesmí obsahovat mezery

Filtr jména elementu

Problém se jménem složky nebo souboru začínajícím číslem nebo interpunkčními znaky, jsem vyřešil jak je vidět z tabulky 2 tak, že před názvy elementů se přidá znak podtržítko „_“. Mezery v názvech elementů jsem jednoduše odstranil. Některé speciální znaky jsou nahrazovány textovou konstantou definovanou aplikací.

Filtr obsahu atributu

Tento filtr pouze nahrazuje nedovolené znaky z obsahu atributu konstantami definovanými v aplikaci.

Tyto filtry jsou použity při indexování a také při zpracování dotazu uživatele, kdy se dotaz musí patřičně upravit, než bude aplikován na kolekci dokumentů.

Aplikace se nezabývá nahrazováním všech nepovolených znaků, pouze znaků ampersand, apostrof, uvozovky, znaménko plus, čárka, podtržítko, složené závorky, hranaté a kulaté závorky. Ostatní nedovolené znaky je možno v budoucnu doimplementovat.

V tabulce 2 lze pozorovat nahrazení znaku „&“ řetězcem „ENT_AMP“ a odstranění mezer u názvu elementu souboru „me ze ra.txt“. Dále si lze všimnout, že pokud se jedná v indexu o soubor má element souboru atribut path, který obsahuje celou cestu k souboru na disku.

Struktura indexu složky, jak je uložena v kolekci v databázi po zaindexování je zachycena v tabulce 2.

Indexovaná složka je: c:\mon

Struktura indexované složky	**Struktura XML dokumentu v indexu**
+mon +1.složka +1. soubor.log +me ze ra.txt +složka +soubor.txt +&.txt	<?xml version="1.0" standalone="yes"?> <index> <_1.složka> <_1.soubor.log path="c:\mon\1.složka\1. soubor.log"/> <_mezera.txt path="c:\mon\1.složka\me ze ra.txt"/> </_1.složka> <_složka/> <_soubor.txt path="c:\mon\soubor.txt"/> <_ENT_AMP.txt path="c:\mon\ENT_AMP.txt"/> </index>

Tabulka 2: Struktura indexu

5.3.2 Prvotní zaindexování

Je-li nově přidávaná složka neprázdná, je nutné její obsah zaindexovat. Bez této první indexace by pak další indexování změn nemělo smysl.

Sedna umožňuje jedním příkazem nahrát validní XML soubor do kolekce v databázi. Toto nahrání je několikanásobně rychlejší (asi 24x, testoval jsem asi na 30000 souborech), než vytváření dokumentu v databázi aktualizačním XQuery jazykem, kdy se pro každý soubor nebo složku, musí vykonat XQuery dotaz, tj. transakce. Za tuto rychlost ovšem zaplatíme tím, že v době nahrávání dokumentu do indexu je celý index uzamčen a nelze s ním pracovat. U obsáhlých složek nahrávání trvá přibližně i 5 minut.

Právě tuto rychlost nahrávání využívá aplikace, kdy po přidání nové složky k indexaci, vytvoří dočasný XML soubor s obsahem stromové struktury složky na disku. Toto vytváření používá dříve uvedené filtry, aby výsledný XML dokument splňoval požadavky správně strukturovaného XML dokumentu a mohl být úspěšně nahrán do indexu.

Jakmile je dočasný dokument (XML soubor) hotov, je jednoduchým příkazem nahrán do indexu jako dokument se jménem id, které bylo vytvořeno hned na začátku přidávání složky do indexu, viz kapitola 5.2.

Jak bylo zmíněno výše, filtr nahrazuje jen některé znaky, proto může vniknout xml soubor, který není správně strukturovaný, bude obsahovat nepovolené znaky. V takovém případě dokument nebude nahrán do indexu a uživatel bude upozorněn dialogem. Celý postup přidání složky do indexu je ilustrován na diagramu 1. Diagram 2 zobrazuje třídní diagram Aplikace pro správa indexů.

5.3.3 Vyhledávání v indexu

Pro vyhledávání v indexu je použit jazyk XPath s malou pomocí XQuery upraveného pro Sednu. Struktura dotazu je následující:

*INDEX/*XPath/XPath/…

Aplikace tento dotaz transformuje na tvar:

*collection('kolekce')/index/*XPath/XPath/*data(@path)*

Na konec dotazu vždy vkládá funkci data(@path). Jejím úkolem je vrátit obsah atributu path elementu který odpovídá hledanému souboru. Tímto vložením také zabraňuje vložení XPath dotazu, který by umožňoval výpis celé struktury indexu. Úkolem je zobrazit absolutní cestu k hledanému souboru a ta je uložena v atributu path. Tímto doplněním funkce data na konec každého dotazu, je nemožné vytvářet složité konstrukce XQuery typu FLWOR. Vzhledem k tomu, že jsou indexovány pouze názvy souborů a složek, je pro dotazování plně dostačující jazyk XPath.

Samozřejmě před spuštěním dotazu aplikace ještě upraví i cesty XPath, tak aby odpovídaly pojmenování elementů v indexu.

Příklad dotazu:

```
INDEX//*[contains(name(.),'jo')]//*[contains(name(.),'compo')]//*[contains(name(.),'feeds')]
```

Tento dotaz vyhledá následující soubory:

- Soubory, které obsahují ve svém názvu řetězec „feeds“ a zároveň
- ve své absolutní cestě obsahují složky (tedy jsou umístěny někde uvnitř těchto složek), které mají ve svém názvu řetězce „jo“ a „compo“ i „feeds“.

Možný příklad výpisu výsledku:

F:\filmy\dama\jom\components\com_newsfeeds\newsfeeds.html.php
F:\filmy\dama\jom\components\com_newsfeeds\newsfeeds.php
F:\filmy\dama\jom\administrator\components\com_newsfeeds\admin.newsfeeds.php
F:\filmy\dama\jom\administrator\components\com_newsfeeds\newsfeeds.class.php
F:\filmy\dama\jom\administrator\components\com_newsfeeds\newsfeeds.xml
F:\filmy\dama\jom\administrator\components\com_newsfeeds\toolbar.newsfeeds.html.php
F:\filmy\dama\jom\administrator\components\com_newsfeeds\toolbar.newsfeeds.php
F:\filmy\dama\jom\administrator\components\com_newsfeeds\admin.newsfeeds.html.php

Malou změnou dotazu:

```
INDEX//*[contains(name(.),'jo')]/*[contains(name(.),'compo')]//*[contains(name(.),'feeds')]
```

Dostaneme jiný výsledek:
F:\filmy\dama\jom\components\com_newsfeeds\newsfeeds.html.php
F:\filmy\dama\jom\components\com_newsfeeds\newsfeeds.php

Zpřesnili jsme zadání tak, že složky obsahující v názvu „jo“ a „compo“ musí následovat za sebou, neboli složka obsahující v názvu „compo“ je uvnitř složky obsahující v názvu „jo“.
Tím jsme zmenšili irelevantní množinu výsledků.

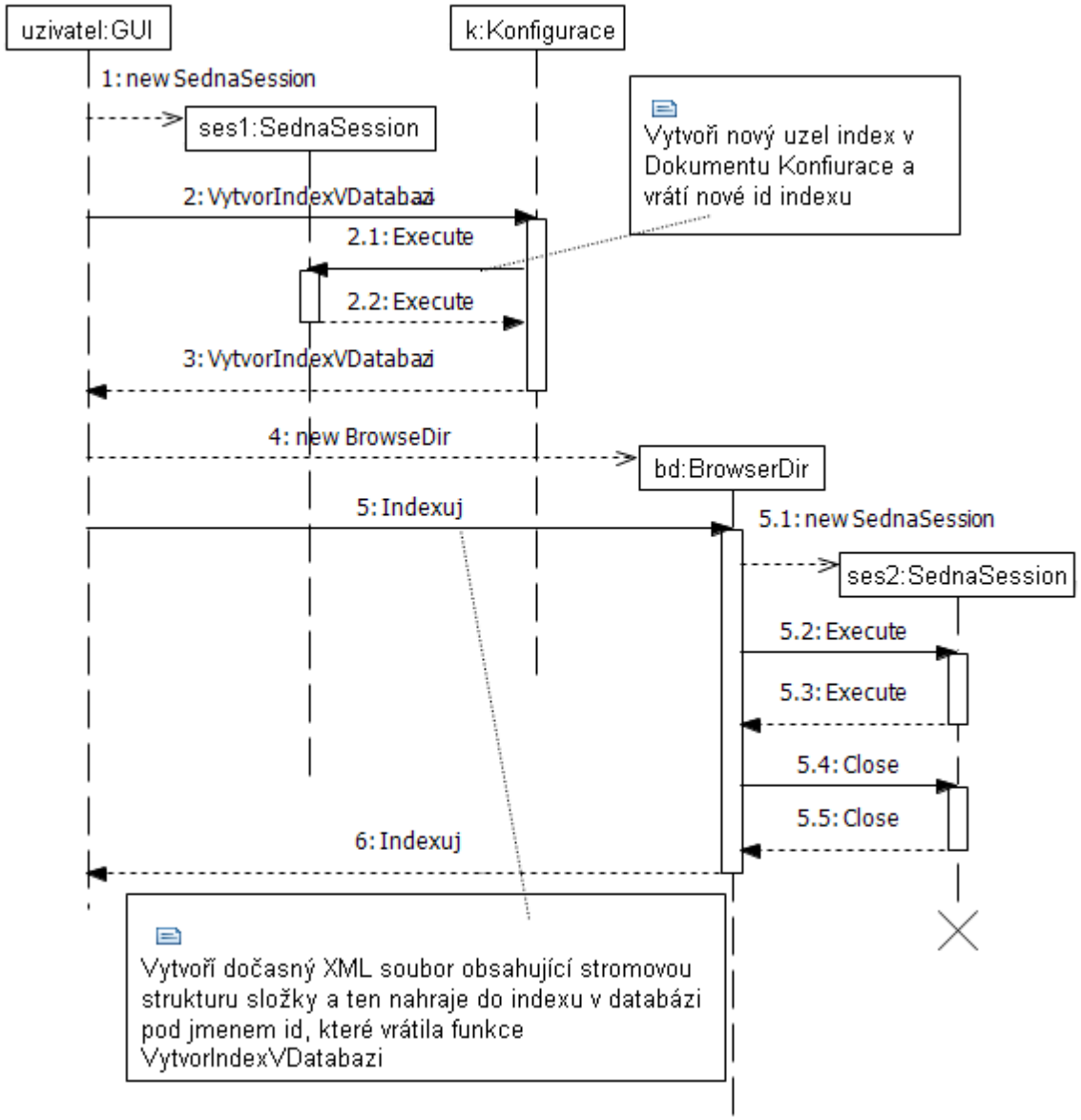


Diagram 1: Prvotní index

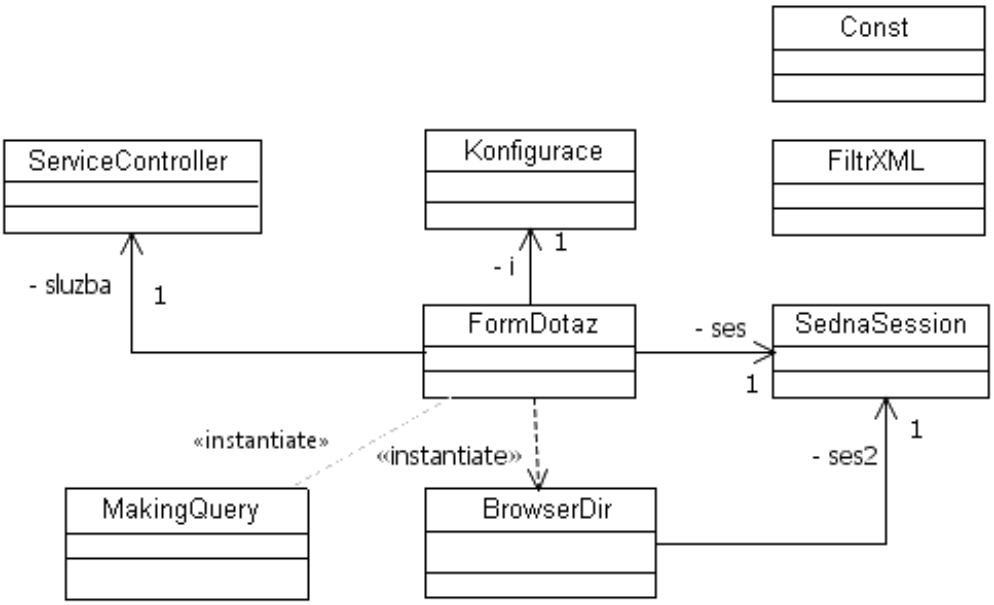


Diagram 2: Třídní diagram Správa indexů

5.3.4 Použití indexů Sedny

Sedna podporuje indexování obsahu elementů a hodnot atributů založené na B-stromu.

Vytváření indexu se provede následujícím příkazem:

```
CREATE INDEX nazev-indexu
ON cesta1 BY cesta2
AS typ
```

Tento příkaz vytvoří index na uzlech specifikovaných cestou „cesta1“ podle klíčů daných cestou „cesta2“.

„Cesta1“ je jednoduchá XPath cesta (bez použití filtrů), která identifikuje uzly dokumentu nebo kolekce, které mají být indexovány. „Cesta2“ je jednoduchá XPath cesta (bez použití filtrů), která specifikuje relativní cestu k uzlům, jejichž textové hodnota jsou použity jako klíče k identifikaci uzlů vrácených cestou „cesta1“. „Cesta2“ nesmí začínat znakem „/“ neno „//“

„Nazev-indexu“ je název indexu a měl by být pro každý index jedinečný.

Typ určuje jaká typová hodnota je přiřazena klíči, může nabývat následujících hodnot:

- xs:string,
- xs:integer,
- xs:float,
- xs:double,
- xs:date,
- xs:time,
- xs:dateTime,
- xs_yearMonthDuration,
- xs_dateTimeDuration

Příklad:

```
CREATE INDEX "lide"
ON doc("auction")/bydliště//osoba BY adresa/mesto
AS xs:string
```

V tomto indexu jsou lidé indexování podle jejich názvu měst jejich bydliště. Pro generování klíčů, v tomto případě názvů měst, je použit typ xs:string.

Jelikož jsme se s vedoucím práce dohodli na struktuře indexu danou v kapitole 5.3.1, neumožňuje rozumně využít indexování Sedna XML databáze. V dané struktuře lze těžko najít klíče, podle kterých by se indexování provádělo.

5.3.5 Aktivace (deaktivace) indexování

Aktivaci indexování službou je nutné provést ručně, jelikož prvotní indexování složky trvá určitou dobu, nemůže být spuštění indexování spuštěno ihned. Služba by totiž mohla pracovat nad nekompletním indexem. To je také důvod, proč při prvotním indexování složky nelze s grafickým uživatelským rozhraním správy indexu pracovat.

Aktivace (Deaktivace) se provede tak, že Aplikace pro správu indexů v dokumentu Konfigurace nastaví hodnotu elementu aktivní na jedničku (nulu) u požadovaného indexu. Po dokončení této změny prostřednictvím komponenty ServiceControler a její funkcí ExecuteCommand pošle číselný parametr běžící službě. Aktivaci ilustruje diagram 3. Průběh aktivace indexování službou je popsána v kapitole 5.5.1.

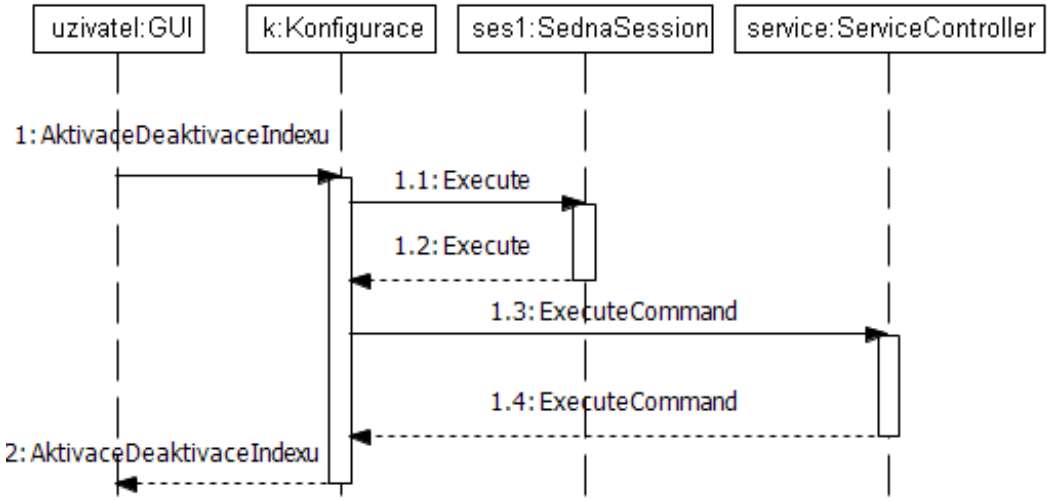


Diagram 3: Aktivace indexování

5.3.6 Problémy indexování systémových složek

Systémové složky, jako například složka System Volume Information, kde si Windows XP, ukládají body obnovení operačního systému indexovat nelze. Operační systém brání aplikaci v přístupu do této složky. To je v aplikaci ošetřeno výjimkou, která přeskočí indexování této složky a pokračuje indexováním složky následující.

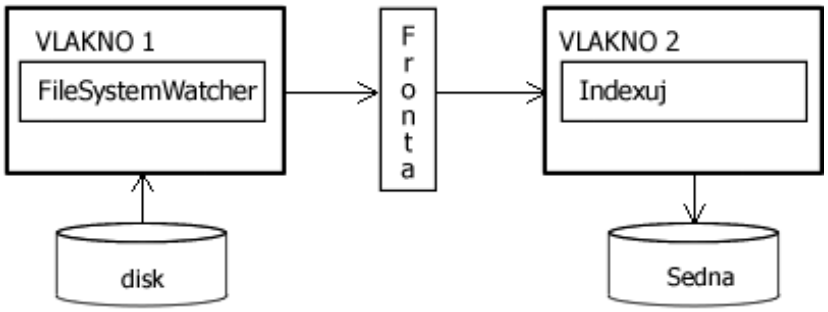
5.4 Zachytávání událostí

Proto, aby mělo indexování smysl a pracovalo správně, je potřeba neustále index aktualizovat, udržovat index integritní s diskovým stromem. To znamená, aktualizovat index při jakékoli diskové operaci, která se týká indexované složky (disku).

Právě proto, je nutné k Aplikaci správy indexu implementovat službu indexování. Jejím úkolem je sledovat změny v indexované složce a tyto změny přenášet do indexu. To byl také hlavní důvod, proč jsem si pro implementaci služby vybral Microsoft .NET Framework a jazyk C#. Prostředí .NET obsahuje spoustu již napsaných knihoven a tříd, nevyjímaje tříd pro práci se soubory a třídy pro sledování změn souborového systému. Dalším důvodem bylo snadný převod aplikace na službu, která poběží na pozadí.

Microsoft. NET Framework nabízí pro sledování změn na disku třídu FileSystemWatcher, která naslouchá změnám disku. Změnami je myšleno vytvoření, odstranění nebo modifikace složky, či souboru. Na základě těchto změn pak vyvolá příslušnou událost. Mým úkolem bylo, naimplementovat zpracování těchto vyvolaných událostí.

Indexování, tedy zpracování jedné události, trvá čas, který Sedna potřebuje k úpravě dokumentu v databázi tedy provedení XQuery dotazu, který je službou na základě změny na disku vytvořen a prostřednictvím Sedna API poslán Sedně. Trvání dotazu je závislé také na velikosti databáze (indexu). Doba provedení dotazu je mnohem delší, než doba provedení operace na disku. Ukázalo se, že v případě kopírování mnoha malých souborů do indexované složky, FileSystemWatcher sice zaregistruje všechny změny na disku a uloží si je do omezeného bufferu, ale velice rychle dojde k přeplnění bufferu a to má za následek, že nejsou brány v úvahu všechny změny, ale jen ty které se zrovna vešli do bufferu. Přeplnění je způsobeno onou delší dobou zpracování dotazu.



Obrázek 6: Řešení pomocí vláken a fronty

Toto přeplňování bufferu bylo pro mou službu nepřípustné, situaci jsem vyřešil pomocí vláken a fronty. Jedno vlákno registruje změny na disku a ukládá je do fronty a druhé vlákno jednotlivé události vybírá z fronty a zpracuje samotné indexování. Situaci ilustruje obrázek 6.

Podrobněji práci vláken popisuje diagram 4.

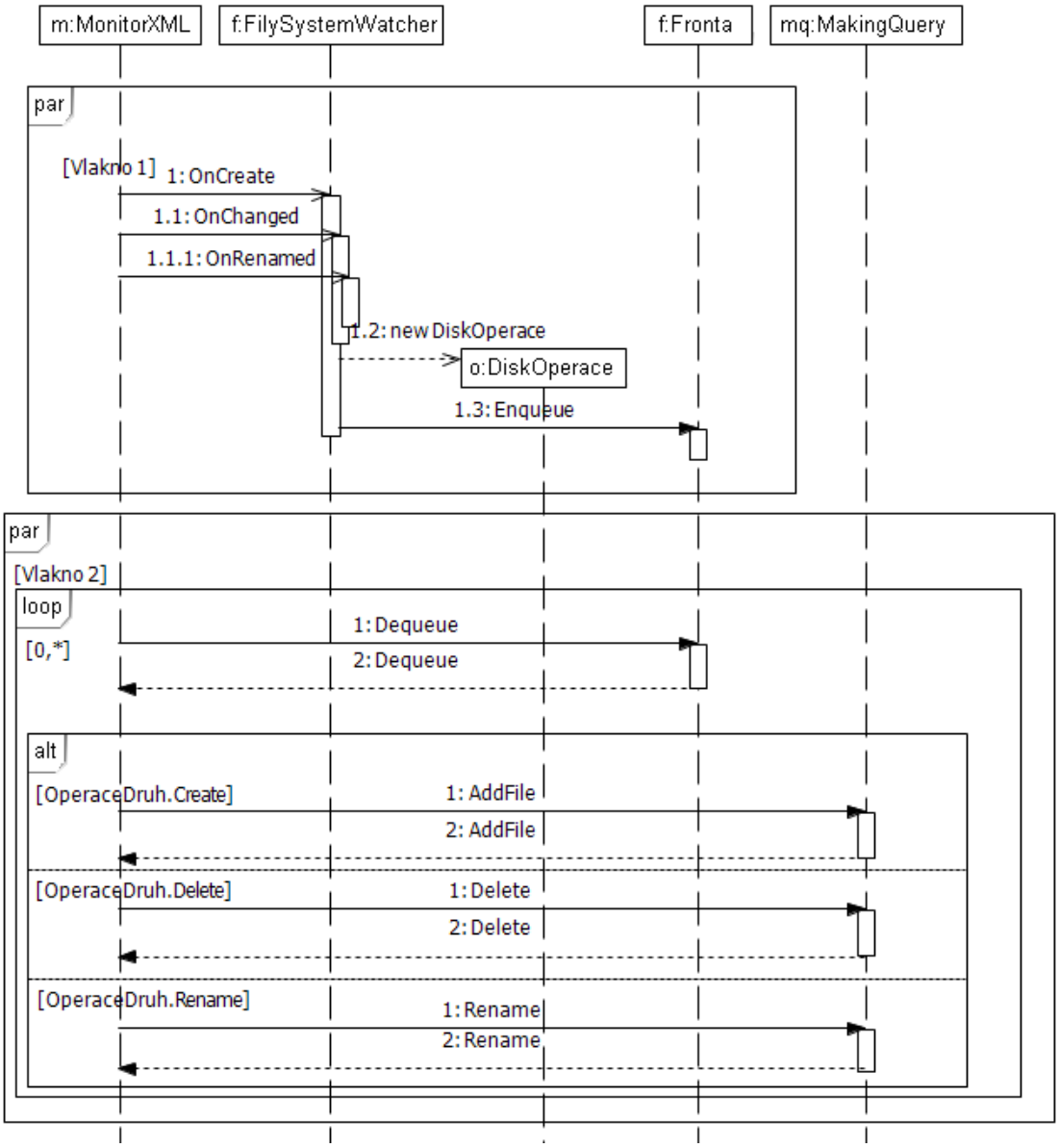


Diagram 4: Práce vláken

Vlákno 2 vybere operaci z fronty a podle druhu operace a jejich parametrů sestaví XQuery dotaz a ten pošle ke zpracování Sedně. Pokud je fronta prázdná, vlákno 2 přejde do suspendovaného stavu. Do akčního stavu ho uvede Vlákno 1 s další novou první události.

Z diagramu 4 je jasné, že v každé instanci třídy MonitorXML běží dvě vlákna s vlastní frontou. Pro každou indexovanou složku vytvoří služba nový objekt typu MonitorXML v němž pracují dvě vlákna.

Nutno dodat, že pokud nově vzniklý nebo změněný soubor, či složka obsahují v názvu nepodporované znaky, které projdou filtrem, tedy nejsou přípustně nahrazeny, sedna hlásí samozřejmě chybu. Tato chyba je zachycena výjimkou, indexace tohoto souboru nebo složky je přeskočena a pokračuje se další položkou ve frontě, jinak by došlo k pádu aplikace.

5.5 Služba indexování souborů

Samotnou službu jsem neimplementoval přímo, jelikož přímá implementace služby neumožňuje ladění a hledání chyb a samotné jednoduché spuštění. Prvním krokem bylo vytvoření běžné aplikace příkazového řádku. Tato aplikace byla testovací a měla pevně nastavené parametry, tedy jedinou indexovanou složku, tzn., byla vytvořena jen jedna instance objektu typu MonitorXML. Teprve, až tato aplikace pracovala správně, přistoupil jsem k převodu aplikace na windows službu. Statickou strukturu služby naznačuje diagram 7.

Převod aplikace na službu

Pomocí Microsoft Visual Studia 2008 stačí vytvořit nový projekt jako službu, vývojové prostředí vyrobí kostru služby. Tedy třídu služby s deklarací povinných metod OnStart() a OnStop(). Mým úkolem bylo implementovat do služby požadovanou funkčnost. Požadovanou funkčností se rozumí chování služby při jejím spuštění, zastavení a navíc aktivaci a deaktivaci indexování na vybrané složce, k čemuž slouží metoda OnCustomCommand().

Aby se služba mohla zaregistrovat do systému Windows je ještě potřeba implementovat instalátor služby. To se provede opět pomocí Visual Studia, které opět vytvoří vše potřebné.

Vytvořený instalátor (třídu) jsem jen jemně modifikoval změnou následujících parametrů:

- jméno služby – Aplikace pro indexování souborů
- účet, pro který se služba smí spustit, v mém případě jsem nastavil lokální systém
- a mód spouštění služby, protože Sedna neumožňuje automatické spouštění jako služba na pozadí, nastavil jsem hodnotu na manuální spouštění, aby se předešlo spuštění služby Aplikace pro indexování souborů, pokud Sedna není aktivní.

Služba se při spuštění připojí k Sedně, z dokumentu Konfigurace načte informace o indexovaných složkách a podle získaných parametrů vytvoří instance objektů MonitorXML. Přesněji pro indexy, které v elementu aktivní obsahují hodnotu jedna, služba vytvoří instanci třídy MonitorXML. Pro možnost aktivace a deaktivace indexování jsou instance objektu typu MonitorXML ukládány do objektu Dictionary, kde klíčem k objektu uloženém v Dictionary je id dokumentu indexované složky. Postup startu služby popisuje diagram 5.

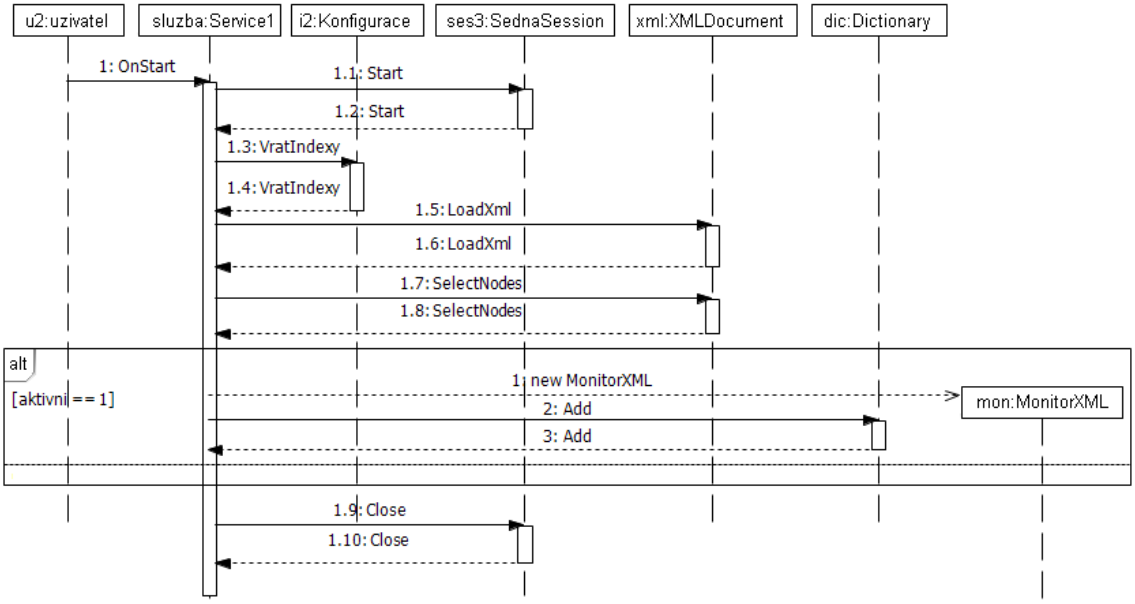


Diagram 5: Spuštění služby

Při zjišťování, zda je indexování na složce aktivní, jsem použil objekt XMLDocument, který je součástí .NET Frameworku. Funkcí LoadXML do něj nahrál dokument Konfigurace obsahující informace o indexovaných složkách. Pak již jednoduchým XPath dotazem získal jen ty složky, které mají aktivní indexování.

5.5.1 Aktivace a deaktivace indexování

Jakmile uživatel v aplikaci správa indexů aktivuje či deaktivuje indexování na vybrané složce (kapitola 5.3.5), posílá aplikace prostřednictvím funkce ExecuteCommand číselný parametr, pomocí kterého služba rozhodne, kterou operaci provést. Při volání ExecuteCommand dojde ve službě ke spuštění funkce OnCustomCommand, jejímž parametrem je právě parametr funkce ExecuteCommand (diagram 6). Uvnitř těla funkce OnCustomCommand se rozhodne, jaký kód bude dále vykonáván. V mém případě si služba z databáze z dokumentu Konfigurace zjistí konfiguraci indexů. V případě aktivace vloží novou instanci objektu typu MonitorXML do objektu Dictionary. V opačném případě je indexování odpovídající složky zastaveno a odpovídající objekt MonitorXML je smazán z Dictionary.

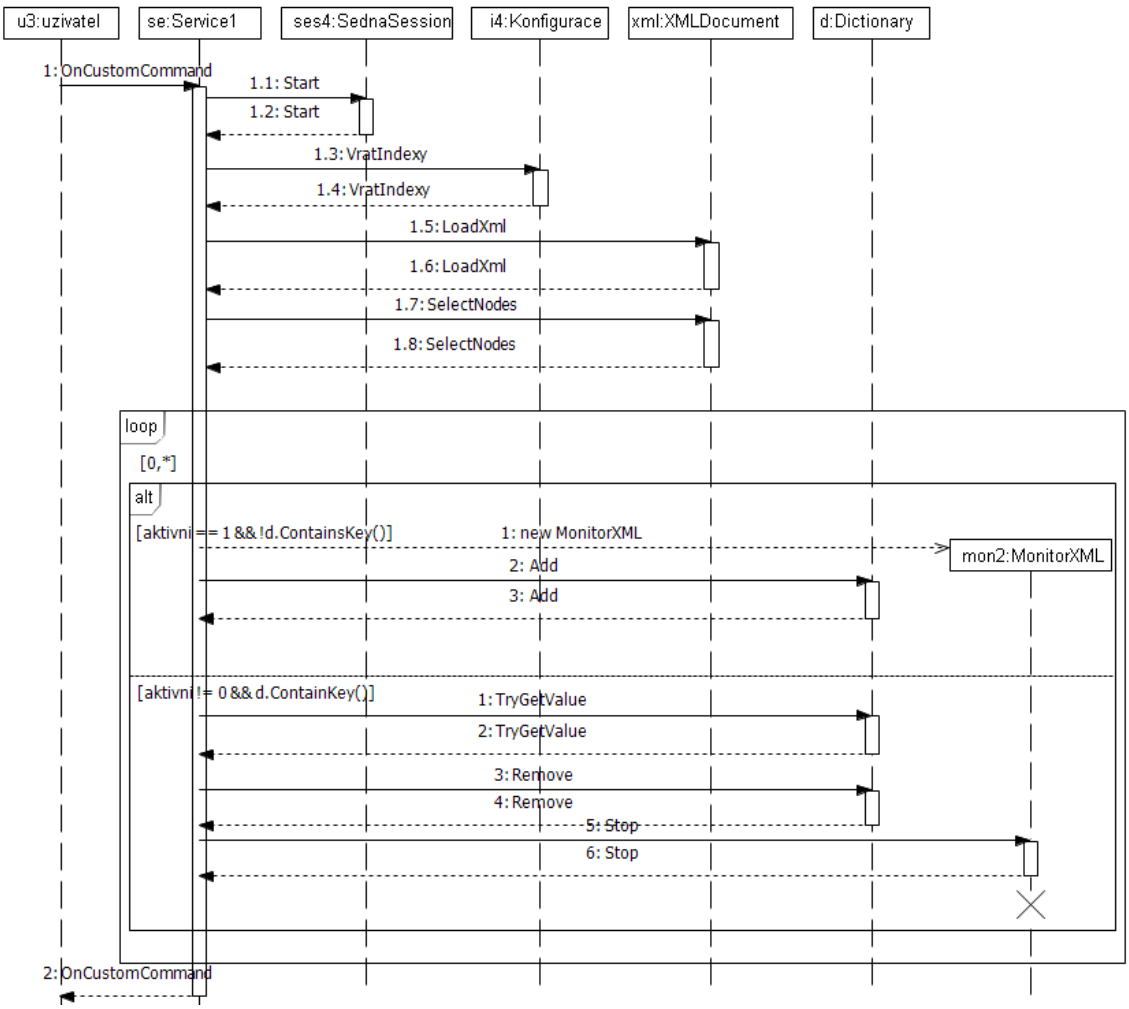


Diagram 6: Aktivace indexování službou

5.5.2 Zastavení služby

Zastavíme-li službu, dojde k přerušení všech pracujících vláken a odstranění všech objektů MonitorXML z objektu Dictionary a ukončení služby.

5.6 Popis tříd Aplikace pro správu indexů a Služby pro indexování souborů

Protože některé třídy jsou používány jak Aplikací pro správu indexů tak Službou indexování souborů, rozhodl jsem se je popsat v jedné shrnující kapitole. Které třídy jsou společné, lze vyčíst z třídních diagramů Aplikace pro správu indexů (diagram 2) a Služby indexování souborů (diagram 7).

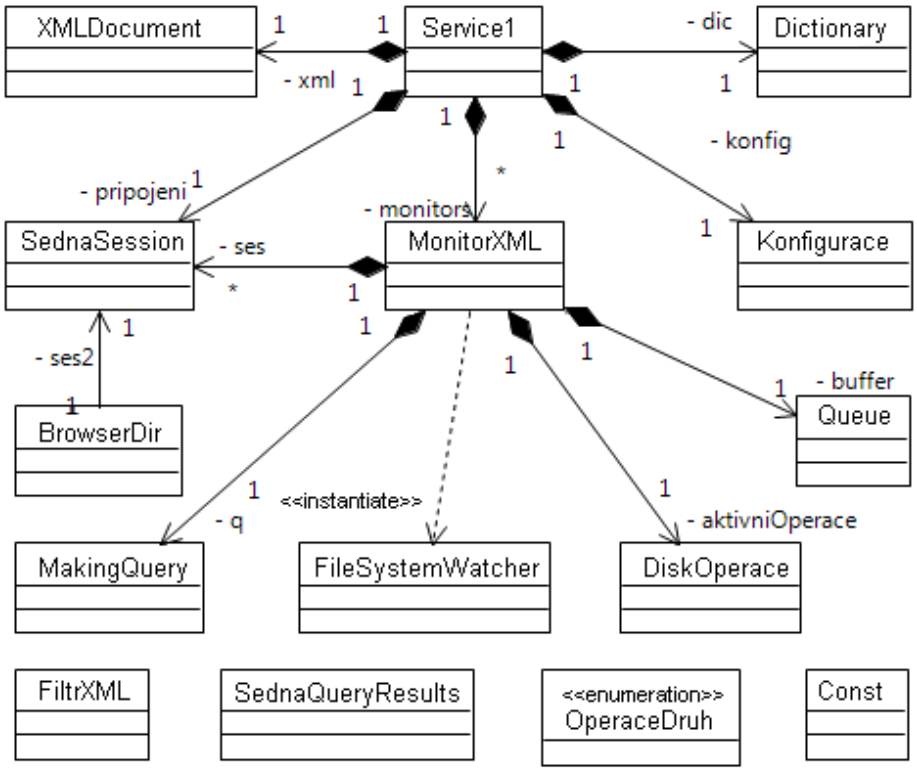


Diagram 7: Třídní diagram služby indexování souborů

5.6.1 třída FormDotaz

Hlavní třída Aplikace pro správu indexů, instance této třídy má kromě zobrazení uživatelské rozhraní následující úkoly:

- Ihned po startu ověřuje, zda neexistují dočasné soubory vytvořené při prvotním indexování, pokud ano jsou smazány
- Připojí se ke správci služeb a zjistí jaký je stav služby indexování souborů, znemožňuje spuštění služby, pokud není aktivní Sedna XML databáze
- Pokusí se připojit k Sedna XML databázi (dále databáze), pokud se připojení nezdaří, znamená to, že databáze nebyla spuštěna, pokusí se tedy databázi spustit. Po úspěšném spuštění databáze, spouští službu indexování souborů.
- Dále umožňuje ovládání aplikace, vyhledávání v indexu, tyto možnosti jsou podrobněji je popsány v příloze uživatelská dokumentace

5.6.2 třída Konfigurace

Tato třída prostřednictvím jazyka XPath a aktualizačního jazyka Sedny zajišťuje následující operace nad konfiguračním XML dokumentem konfigurace uloženým v XML databázi:

- Poskytuje aplikaci informace o aktuální konfiguraci indexů
- Aktualizuje stav indexu, zda je indexování složky aktivní či nikoliv
- Pokud prostřednictvím Aplikace pro správu indexů přidáváme novou složku k indexaci, vytváří nový uzel index v dokumentu konfigurace s odpovídajícími hodnotami nově indexované složky.

5.6.3 třída BrowserDir

Aplikace používá tuto třídu k následujícím operacím:

- prvotní zaindexování složky do dočasného XML souboru a jeho následné nahrání do databáze
- Odstranění indexované složky z indexu
- Znovu zaindexování přejmenované neprázdné složky

- Pokud přejmenujeme neprázdnou složku uvnitř aktivně indexované složky, pak informace v atributu path souborů uvnitř této složky jsou neplatné a je potřeba je znovu zaindexovat.

5.6.4 FiltrXML

Obsahuje statické funkce

- XmlTextElement – nahrazuje vybrané nepřípustné znaky v názvu elementu
- XmlTextAttribut – nahrazuje vybrané nepřípustné znaky v obsahu atributu
- XmlTextPath – dotaz uživatele musí být patřičně upraven, tak aby odpovídal struktuře indexu, navíc může také obsahovat nepřípustné znaky, které budou opět nahrazeny.

Nepřístupné znaky jsou nahrazovány řetězci, které jsou definovány uvnitř třídy Const.

Tyto zajišťují, že jsou elementy správně pojmenované a obsah atributu path obsahuje přípustné znaky. Teprve pak mohou být elementy přidány do indexu (který je XML dokumentem a je tedy správně strukturovaný a splňuje specifikaci W3C).

5.6.5 třída Const

Uvnitř této třídy jsou definovány následující konstanty:

- Informace pro připojení k Sedna XML databázi (host, port, jméno databáze, uživatel, heslo)
- Jméno kolekce, do které jsou ukládány všechny indexy složek
- Velikost bufferu pro zachytávání událostí
- Konstanty nahrazující speciální znaky používané třídou FiltrXML

5.6.6 třída MakingQuery

Vytváří syntakticky správné dotazy pro Sedna XML databázi. Upravuje vložený dotaz uživatele tak, aby odpovídal struktuře indexu, a tento dotaz vykoná a vrací výsledek.

Z hlediska indexování sestaví a vykoná operace pro událost:

- Vznik nové složky nebo souboru
- Smazání složky nebo souboru
- Přejmenování složky nebo souboru

5.6.7 třída MonitorXML

Hlavní třída Služby indexování souborů určená k monitorování změn uvnitř indexované složky. Používá třídu FileSystemWatcher Microsoft.NET frameworku. Na základě těchto změn volá příslušné metody, které aktualizují stav indexu.

Pro každou indexovanou složku je službou vytvořena instance této třídy.

5.6.8 třída Queue

Třída Microsoft.NET frameworku. Slouží jako vyrovnávací paměť (buffer), pro uložení zatím neobsloužených událostí (změn v indexované složce).

5.6.9 struktura DiskOperace

Obsahuje informace o druhu operace vyvolané změnou v indexované složce, které jsou nutné pro správnou aktualizaci indexu. Například v případě přejmenování souboru obsahuje:

- druh operace = přejmenování (Rename)
- absolutní cestu k souboru
- staré jméno souboru
- nové jméno souboru

5.6.10 výčtový typ (enum) OperaceDruh

Obsahuje tři prvky

- Create (Vytvořit)
- Rename (Přejmenovat)
- Delete (Smazat)

5.6.11 třída XmlDocument

Třída Microsoft.NET frameworku. Reprezentuje XML dokument.

5.6.12 třída SednaSession

Třída Sedna.NET API umožňující navázaní spojení a vykonávání dotazů Sedna XML databáze.

5.6.13 třída SednaQueryResults

Třída Sedna.NET API. Obsahuje výsledky XPath/XQuery dotazu.

5.6.14 třída Dictionary

Třída Microsoft.NET frameworku. Služba využívá tuto třídu k ukládání aktivních instancí třídy MonitorXML. Každá indexovaná složka má své jedinečné identifikační číslo (id). Každé indexované složce je její monitor uložen v Dictionary pod klíčem jejího id. V případě deaktivace indexování na složce je její monitor podle id nalezen, zastaven a odstraněn.

5.6.15 třída Service1

Třída služby indexování souborů. Má implementovány metody:

- OnStart() – vykoná se při spuštění služby, zjišťuje konfiguraci indexovaných složek, spouští indexování na složkách
- OnStop() - vykoná se při zastavení služby, zastavuje veškeré indexování
- OnCustomCommand(int) – vykoná operaci danou číselnou hodnotou parametru, je vyvolána Aplikací pro správu indexů při aktivaci a deaktivaci indexování složky.

6 Implementace

Navržené architektury aplikace a služby jsou realizovány prostřednictvím programovacího jazyka C sharp a Microsoft.NET frameworku (prostředí ve kterém se vyvíjejí aplikace). Jako vývojové prostředí bylo použito Microsoft Visual Studio 2008.

Aby bylo možno používat některé třídy jak Aplikací pro správu indexů, tak Službou indexování souborů, jsou některé skupiny tříd kompilovány jako samostatné assembly. Aplikace a služba pak mohou sdílet tyto assembly. Rozdělení tříd do jednotlivých assembly ukazuje diagram 8. Výhodou tohoto rozdělení je modularita aplikace a služby.

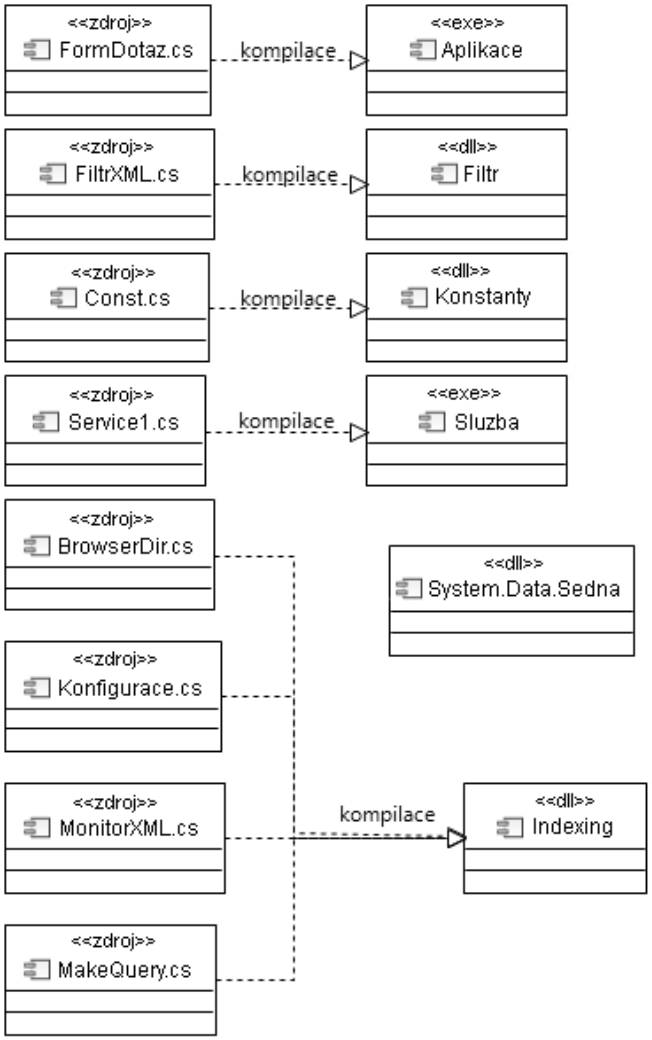


Diagram 8: Implementace tříd do assembly

Jak bude zmíněno níže služba a aplikace, při indexování souborů a složek, které v názvu obsahují nepřípustné znaky, nahrazuje tyto znaky konstantami. Pro toto nahrazování používají třídy:

- Const – definovány konstanty, kterými jsou nahrazeny nepřípustné znaky
- FiltrXML – obsahuje funkce, které vykonávají toto nahrazení

V rámci této práce jsem nemohl zahrnout celou množinu nepřípustných znaků. Pokud by ovšem v budoucnu bylo žádoucí zahrnout celou množinu nebo množinu nepřípustných znaků zvětšit, stačí provést tyto kroky:

- Ve třídě Const vytvořit nové konstanty pro další nepřípustné znaky
- Modifikovat funkce ve třídě FiltrXML, tak aby při nahrazování zahrnuly i nově definované konstanty
- Zkompilovat tyto dvě třídy, tím vzniknou nové assembly (Filtr.dll, Konstanty.dll)
- Nově vzniklými assembly nahradit ty stávající

Těmito kroky jsme rozšířili množinu nepřípustných znaků pro aplikaci i službu zároveň, bez nutnosti kompilace celé aplikace a služby.

6.1 Sedna.NET API

Pro komunikaci se Sednou XML databází je použito programové rozhraní Sedna.NET, které je tvořeno jednou assembly s názvem System.Data.Sedna, obsahující 10 tříd. V přímé implementaci v aplikaci jsou použity dvě a to:

- třída SednaSession – obsahuje funkce pro navázání/uzavření spojení se sednou, vykonávání transakcí
- třída SednaQueryResults – obsahuje kolekci výsledků dotazů, je implementována jako enumerátor.

Toto rozhraní je jedno ze dvou existujících, které mi bylo doporučeno Sedna Týmem.

7 Testování

Na výsledky testování má velký vliv použité API, prostřednictvím kterého Sedna komunikuje s aplikací a službou. Tam kde Sedna vydává výsledky s velkou prodlevou, skončí operace vyvoláním výjimky. Bohužel k dispozici byly jen dvě nepodporovaná API pro programovací prostředí NET. Použité API mi bylo doporučeno Sedna Týmem, se kterým jsem prostřednictvím fóra řešil problémy ohledně API.

Testování je prováděno na indexech třech velikostí, vždy pro operaci vytváření indexu složky, vyhledávání v indexu, aktualizace a odstranění indexu složky. Výsledek je pro srovnání zaznamenán pro každou operaci v tabulce. Obsah a označení testovaných indexů je uveden v tabulce 3.

Obsah indexu	Velikost MB	Hloubka stromu	Označení
30 souborů ve 3 složkách	10	3	INDEX - 1
3536 souborů ve 422 složkách	35	6	INDEX - 2
40 652 souborů ve 4174 složkách	968	6	INDEX - 3

Tabulka 3: Označení indexu

Rychlost provádění operací Sedny jsem zjišťoval prostřednictvím protokolu (textový soubor) událostí, který si tato databáze vytváří. Ukázku lze najít v příloze.

7.1 Vytvoření prvotního indexu

Rychlost vytvoření prvotního indexu Aplikací pro správu indexů je závislá na tom kolik složek a souborů indexovaná složka obsahuje. Pokud indexovaná složka obsahuje v názvech vnořených složek a souborů nepřípustné znaky prvotní index nebude vytvořen. Tento problém vysvětluje kapitola 5.3.2. Pro srovnání jsem testy provedl na velmi malém indexu a velkém indexu, sledoval jsem rychlost zaindexování a rychlost růstu databázového soboru. Z tabulky 4 je patrné, že pro velmi velké indexy již nemá cenu testovat, protože velikost databázového souboru roste velmi rychle.

index	čas	velikost datového souboru
INDEX - 1	3 sekundy	100 MB
INDEX - 2	7 sekund	700 MB
INDEX - 3	3 minuty	8 GB

Tabulka 4: Vytvoření prvotního indexu

INDEX - 1

- čas vytvoření indexu je zanedbatelný, v tomto případě nedošlo ani ke zvětšení databázového souboru z jeho základní velikosti, která je 100 MB. Podrobně se problematice zvětšování datových souborů věnuje kapitola 3.4.

INDEX - 2

- čas vytvoření tohoto indexu trval relativně krátce
- databázový soubor se rychle zvětšuje

INDEX - 3

- čas vytvoření indexu je v tomto případě již v řádu minut.
- velikost databázového souboru je však obrovská, zabere více místa než samotné indexované sobory

7.2 Vyhledávání v indexu

Rovněž vyhledávání Sedny v indexu velmi závisí na jeho velikosti. Výpis většího počtu výsledků i v malém indexu ovšem trvá dobu delší, to je bohužel dané způsobem implementace prezentování výsledků v Sedna.NET API, které se nakonec ukázalo jako velmi nekvalitní.

Testování jsem prováděl na čtyřech testovacích XPath dotazech, kde jeden byl pro nejhorší případ, kdy se musí projít celý index. Testovací XPath dotazy jsou uvedeny v tabulce 5.

Q-1	**Dotaz**	*INDEX//*[contains(name(.),'x')]*
	Popis	x je znak nebo řetězec, který má soubor v názvu obsahovat, musí se prohledat kompletně celý index
Q-2	**Dotaz**	*INDEX/golf/*[contains(name(.),'x')]*
	Popis	v tomto případě, známe kus cesty, prohledána je část indexu omezena složkou golf
Q-3	**Dotaz**	*INDEX/*[contains(name(.),'jo')]//*[contains(name(.),'version')]*
	Popis	známe obsah části názvu složek, prohledávají se uzly na nejvyšší úrovni indexu obsahující v názvu slovo „jo“
Q-4	**Dotaz**	*INDEX/sanglab[.//fonts]/admin/login[.//forgot]/*[contains(name(.),'war')]*
	Popis	Strom složky „sanglab“ musí někde v podstromu obsahovat složku „fonts“, pak v pod-složce „admin“ a pod-složce „login“ najdi soubory obsahující řetězec „war“, přitom složka „login“ musí obsahovat pod-složku s názvem „forgot“

Tabulka 5: Testovací dotazy

INDEX - 1

- Dotazy **Q-1 až Q-4 –** výsledky dotazů byly k dispozici do jedné sekundy. Vyhledávání Sedny v takto malém indexu je rychlé, přístup na disk je minimální, většina databázového souboru je totiž umístěna v paměti. I po zvětšení indexu na obsah 800 souborů v 38 složkách rychlost zůstala stejná.

INDEX -2

- **Dotazy Q-1 až Q-4** - stejně jako v předchozím případě všechny dotazy proběhly do jedné sekundy

INDEX - 3

- Dotaz **Q-1** (prohledává se celý index). Lze pozorovat velké využití disku, Sedna XML databáze stále pracuje v prohledávání indexu, avšak zbytečně, protože díky Senda.NET API je již aplikace ukončena, protože Sedna dlouho neodpovídala**.** Vyhledávání v celém takto velkém indexu je neúspěšné
- Dotazy **Q-2 až Q-4** (vyhledávání je omezeno na nějakou složku nebo více podmínkami), jsou provedeny rychlostí do jedné sekundy.

7.3 Aktualizace indexu

Aktualizaci indexu již provádí Služba indexování souborů. Na základě provedené operace uvnitř indexované složky zašle požadovaný aktualizační příkaz XML databázi. Z tabulky 6 je zřejmé, že pro všechny aktualizační operace ve všech testovaných indexech Sedna zvládne dotaz vykonat stejně rychle.

index	Vytvoření souboru/složky [sekund]	Přejmenování souboru/složky [sekund]	Odstranění souboru/složky [sekund]	Přibližné zvětšení datového souboru
INDEX - 1	< 1	< 1	< 1	+200 MB
INDEX - 2	< 1	< 1	< 1	+ 200 MB
INDEX - 3	< 1	< 1	< 1	+ 200 MB

Tabulka 6: Aktualizace indexu

INDEX - 1

- Při vytvoření nové prázdné složky je rychlost aktualizace menší než sekunda
- Při nakopírování zhruba 800 nových souborů ve 38 složkách do indexované složky již indexování trvalo přibližně 5 minut a datový soubor se zvětšil na 200 MB. Za 1 sekundu tak Sedna stihne zaindexovat tři soubory.
- Pro smazání 100 souborů v 3 složkách Sedna potřebovala k odstranění odpovídajících elementů přibližně 30 sekund, odpovídá třem odstraněným elementům za sekundu.

- Přejmenování neprázdné složky vyžaduje nahrazení celého elementu složky novým prázdným elementem s novým názvem a znovu zaindexování jejího obsahu. Opět je rychlost indexování 3 soubory na jednu sekundu.
- Po provedení všech těchto operací byla konečná velikost datového souboru 300 MB.

INDEX - 2

- rychlost aktualizace byla stejná jako v předchozím případě pří velikosti databázového souboru necelý jeden GB

INDEX - 3

- i v případě velikosti databázového souboru 8 GB, byla rychlost aktualizace jak při vytvoření, tak při mazání, 3 soubory za 1 sekundu

7.4 Odstranění indexu složky

Odstranění indexu složky provádí Aplikace pro správu indexů. Provede se odstraněním příslušného dokumentu indexované složky z indexu (kolekce). Bohužel Sedna v případě odstranění indexu složky zvětší datový soubor celého indexu (z důvodu verzování) o velikost odstraněného indexu této složky a navíc docela hodně zaměstnává pevný disk. Rychlost odstranění a velikost datového souboru je zaznamenána v tabulce 7.

index	čas trvání odstraňování	velikost datového souboru
INDEX - 1	20 sekund	400MB
INDEX - 2	-	(2 GB)
INDEX - 3	-	(16 GB)

Tabulka 7: Odstranění indexu

INDEX – 1 aktualizován přidáním 1000 souborů

- Už u takto malého indexu je čas odstranění docela velký
- Velikost datového souboru se zvětšila na 400 MB, tedy na dvojnásobek. Zvětšování je zásluhou mechanizmu verzování.

Odstranění indexu složky o velikosti přesahující 3 500 souborů v 422 složkách

- už při odstranění takto velkého indexu dojde k chybě, Sedna nestačí API včas odpovědět a API tak ukončí spojení, dojde k pádu aplikace, sedna však stačí databázový soubor zvětšit, vytváří starší verzi
- tady se ukázalo, proč se datový soubor XML databáze při vymazání dokumentu z kolekce zvětšuje o velikost vymazaného dokumentu, je to záloha v případě pádu a neúspěšného dokončení operace.
- při opětovném spuštění aplikace, je použita původní (starší) verze indexu

Odstranění indexu složky o velikosti přesahující 40 652 souborů v 4 714 složkách

- nemělo smysl testovat
- původní soubor by se podle předpokladů zvětšil o 8 GB

8 Závěr

Indexování souborů do zvolené volně dostupné Sedna XML databáze se ukázalo jako pomalé a kapacitně velmi náročné. Sedna vytváří při tvorbě indexu složky, která obsahuje několik tisíc souborů, obrovský datový soubor, jehož kapacita s pozdějšími aktualizacemi a přidáváním dalších indexů složek pouze roste. I v případě odstranění indexu, databázový soubor z důvodu zálohování naroste o kapacitu zabírající právě odstraňovaným indexem. V dnešní době, kdy se na počítačích vyskytují sta tisíce souborů, by bylo nemožné zaindexovat si takto například celý pevný disk, protože by jednoduše došla jeho kapacita.

Vyhledávání v takto vytvořeném indexu však má své výhody. Díky XML struktuře a jazyku XPath, lze v indexu vyhledávat nejen na základě masky souboru. Pokud například ani neznáme jméno souboru, ale matně si vzpomínáme, ve které složce jsme ho naposled viděli, lze ho pomocí takto vytvořeného indexu snadněji vyhledat, zadáním části jména této složky. Získáme tak výpis souborů, které mají ve své absolutní cestě tuto složku.

Sedna se prokázala jako rychlá v případě prohledávání malých indexů (do tisíce souborů, složek), kdy celý datový soubor nebo jeho větší část je načtená v operační paměti. V takto malém indexu nebyl problém nalézt rychle odpověď i v případě nutnosti prohledat celý XML strom indexu, odpověď byla okamžitá.

Jedná-li se ale již o obsáhlý index (40 000 a více souborů), bylo již znát v případě nutnosti prohledat celý strom indexu drastické zpomalení, které spadalo do řádů minut. Jelikož se v mém případě vyskytuje v hlavní paměti „jen“ 600 MB dat z celého 8 Gigabytového datového souboru, Sedna musí číst z disku a nutno podotknout, že ho velmi zatěžovala, takže o rychlém výsledku se nedá hovořit.

Zlepšení rychlosti poskytnutí výsledku lze docílit omezením se na určitou část indexu, uzel složky, vyhledávání bylo rychlé a ani příliš nezatěžovalo disk.

Aplikace indexování souborů měla původně pracovat jako služba spouštějící se automaticky po startu Windows, tento požadavek nebylo možné splnit. V současné době se mi nepodařilo najít XML databázi, která by pracovala jako služba (mimo eXist, ale ten je webovou aplikací),

nejblíže k ní měla právě Sedna, která umožňuje pracovat na pozadí. Vyřešil jsem to tak, že při spuštění Aplikace pro správu indexů se spustí Sedna XML databáze a po ní se spustí Služba indexování souborů, která se při instalaci celé aplikace zaregistruje v systému Windows.

Použité Sedna.NET API, se ukázalo, jako nekvalitní pro práci s velkou databází, naneštěstí jediné, které mi Sedna tým ze dvou existujících doporučil. Při vyhledávání v takto obsáhlých indexech je odezva Sedny dlouhá. Pokud Sedna neodpoví na dotaz v maximálním časovém intervalu, který je nastaven API na 37 sekund, Sedna.NET API vynuceně ukončí spojení se Sednou a následně na to se Aplikace ukončí. Sedna je přitom na dotaz schopná odpovědět, ale nestihne to do 37 sekund. API se projeví i při odstraňování velkého indexu, Sedna opět dlouho pracuje s diskem a nevrací odpověď, API vynuceně ukončí spojení se Sednou a aplikace se ukončí.

Jelikož jsem přidávání, odstraňování a vyhledávání v indexu prostřednictvím aplikace implementoval jako poslední, nemohl jsem tušit, že API takto zklame. V implementaci služby API pracuje spolehlivě.

Pokud by byla použitá jiná XML databáze, která by byla rychlejší, vytvářela datové soubory přijatelnější velikosti, je indexování do XML databáze dobrou cestou jak zrychlit vyhledávání souborů a umožňuje ke specifikaci souboru použít nejen masku, ale i fragmenty cesty k souboru.

Vytvořená aplikace splňuje požadavky zadání, je schopna převést adresářovou strukturu neobsahující nepodporované znaky na XML dokument a tento uložit do XML databáze do kolekce (která je indexem). Běžící služba, monitoruje změny v indexovaných složkách a udržuje tento index integritní. Aplikace umožňuje v tomto indexu vyhledávat pomocí jazyka XPath způsobem popsaným v příloze.

9 Seznam použité literatury a zdrojů

[1] Maxim Grinev, Andrey Fomichev, Sergey Kuznetsov. *Sedna: A Native XML DBMS.* [Dokument] 2004.

[2] eXist. *eXist-db Open Source Native XML Database.* [Online] 2010. http://exist.sourceforge.net/.

[3] Lehti, Patrick. Design and implementation of a data manipulation processor for an XML query processor. Technical report. *Diplomová práce.* Darmstadt, Německo : Technical University of Darmstadt, 2001.

[4] Albrechtová, Zdeňka. Diplomová práce. *Ukládání geodat do XML nativních databází.* Plzeň, Česká republika : Západočeská univerzita v Plzni, 2007.

[5] Tamino | The XML Database – Software AG. *SOA| BPM | ESB | Integration – Improve Business Process Faster with Software AG.* [Online] Software AG, 2010. http://www.softwareag.com/Corporate/products/wm/tamino/default.asp.

[6] Extensible Markup Language (XML). *W3C.* [Online] W3C, 2010. http://www.w3.org/XML/.

[7] Činčala, Radoslav. Bakalářská práce. *Testovanie výkonu existujúcich XML databáz.* 2009.

[8] Query Processing at Light Speed. *Query Processing at Light Speed.* [Online] CWI, 1994-2010. http://monetdb.cwi.nl/.

[9] Community: XML Technologies. *EMC Community Network - ECN.* [Online] EMC, 2010. https://community.emc.com/community/edn/xmltech.

[10] RAS, Institute for System Programming. Sedna XML Database. *Sedna XML Database System.* [Online] 2003-2009. http://modis.ispras.ru/sedna/index.html.

[11] -. *TIMBER.* [Online] 2009. http://www.eecs.umich.edu/db/timber/introduction.html.

[12] XML Database and XPath/XQuery Full Text Processor. *BaseX.* [Online] Christian Grün, DBIS, U Konstanz, 2010. http://www.inf.uni-konstanz.de/dbis/basex/.

[13] *TagSoup - Just Keep On Truckin.* [Online] 2002. http://home.ccil.org/~cowan/XML/tagsoup/.

[14] XSLT 2.0 and XQuery 1.0 Serialization. *W3C Recommendation.* [Online] 2007. http://www.w3.org/TR/xslt-xquery-serialization/.

[15] XQuery Update Facility 1.0. *W3C Candidate Recommendation.* [Online] 2009. http://www.w3.org/TR/xquery-update-10/.

[16] XQuery and XPath Full Text 1.0. *W3C Candidate Recommendation.* [Online] 2010. http://www.w3.org/TR/xpath-full-text-10/#tq-ftsearch-xml.

[17] JAX-RX. *Sourceforge.net.* [Online] 1999-2009. http://jax-rx.sourceforge.net/.

[18] Documentation. *BaseX.* [Online] http://www.inf.uni-konstanz.de/dbis/basex/rest.

[19] jetty. *Jetty WebServer.* [Online] 1995-2009. http://jetty.codehaus.org/jetty/.

[20] The Extensible HyperText Markup Language. *W3C.* [Online] 2002. http://www.w3.org/TR/xhtml1/.

[21] Cascading Style Sheets. *W3C.* [Online] 2010. http://www.w3.org/Style/CSS/.

[22] JavaScript Language Specification. [Online] 1996. http://hepwww.rl.ac.uk/Adye/jsspec11/jsrefspe.htm.

[23] XMLHttpRequest. *W3C.* [Online] 2009. http://www.w3.org/TR/XMLHttpRequest/.

[24] *WebDAV Resources.* [Online] http://www.webdav.org/.

[25] SOAP Specifications. *W3C.* [Online] 2007. http://www.w3.org/TR/soap/.

[26] XML-RPC Specification. *XML-RPC.COM.* [Online] 1999. http://www.xmlrpc.com/spec.

[27] The Atom Publishing Protocol. *BitWorking.* [Online] 2005. http://bitworking.org/projects/atom/draft-ietf-atompub-protocol-06.html.

[28] Document Object Model (DOM). *W3C.* [Online] 1997-2005. http://www.w3.org/DOM/.

[29] *OASIS eXtensible Access Control Markup Language (XACML) TC.* [Online] 1993-2010. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf.

[30] Apache Ant User Manual. *Apache Ant.* [Online] http://ant.apache.org/manual/index.html.

[31] XML Inclusions (XInclude) Version 1.0. *W3C.* [Online] 2006. http://www.w3.org/TR/xinclude/.

[32] XML Pointer Language (XPointer) Version 1.0. *W3C.* [Online] 2001. http://www.w3.org/TR/WD-xptr.

[33] Axis. *WebServices.* [Online] 2000-2005. http://ws.apache.org/axis/.

[34] W3C. XML Path Language (XPath). *World Wide Web Consortium (W3C).* [Online] 2010. http://www.w3.org/TR/xpath/.

[35] XQuery 1.0: An XML Query Language. *W3C.* [Online] 2007. http://www.w3.org/TR/xquery/.

[36] Microsoft. FileSystemWatcher Class (System.IO). *Microsoft MSDN .NET Framework Developer Center*. [Online] Microsoft, 2009. http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher.aspx.

[37] Wiki. *Wiki - PostgreSQL*. [Online] 2009. http://www.postgres.cz/index.php/Slovník.

[38] Wikipedia. [Online] 2009. http://cs.wikipedia.org/wiki/Deklarativní_programování.

10 Přílohy

A. Obsah CD

Obsah adresářů:

- aplikace – instalační soubor aplikace setup.exe
- src – zdrojové soubory aplikace, projekt Microsoft Visual Studia 2008
- text – text diplomové práce
- NET-Framework-35 – Microsoft. NET Framework 3.5
- doc – programátorská dokumentace

B. Uživatelský manuál

I. Instalace

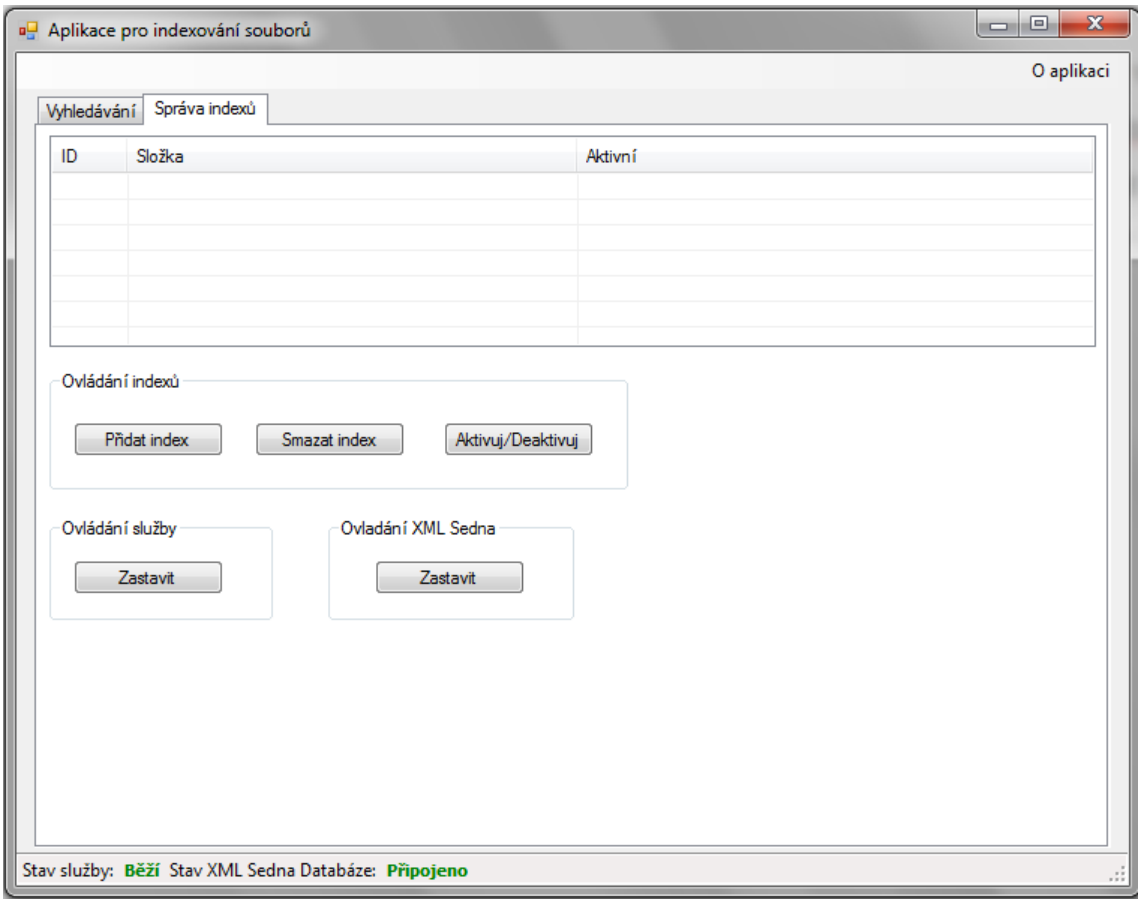
Pro spuštění aplikace je nutné mít nainstalován Microsoft. NET Framework verze 3.5, který je k dispozici na přiloženém CD. Dále potřebuje pro svůj běh 600 MB operační paměti a minimálně 500 MB kapacity na disku, pro testování minimálně 2 GB.

Aplikace je na CD umístěna ve složce aplikace, která obsahuje soubor setup.exe. Pro instalaci stačí soubor spustit a následovat kroky instalátoru. Aplikace je testována pod operačním systémem Windows XP a Windows 7.

Instalační program nainstaluje aplikaci a službu do zvolené složky, vytvoří základní databázi indexu, registruje službu do systému Windows a spustí na pozadí Sedna XML databázi. Průběh instalace může být přerušován firewallem, kde je nutné vždy povolit přístup. Pokud Aplikace nebude fungovat, je blokována firewallem a je potřeba vytvořit výjimky (postup závisí na firewallu).

II. Spuštění aplikace

Zástupce aplikace lze najít v nabídce programy v nabídce start. Aplikace pro správu indexů je zobrazena na obrázku 7. Pro ovládání služby a databáze a správy indexů je určena záložka *Správa indexů*.



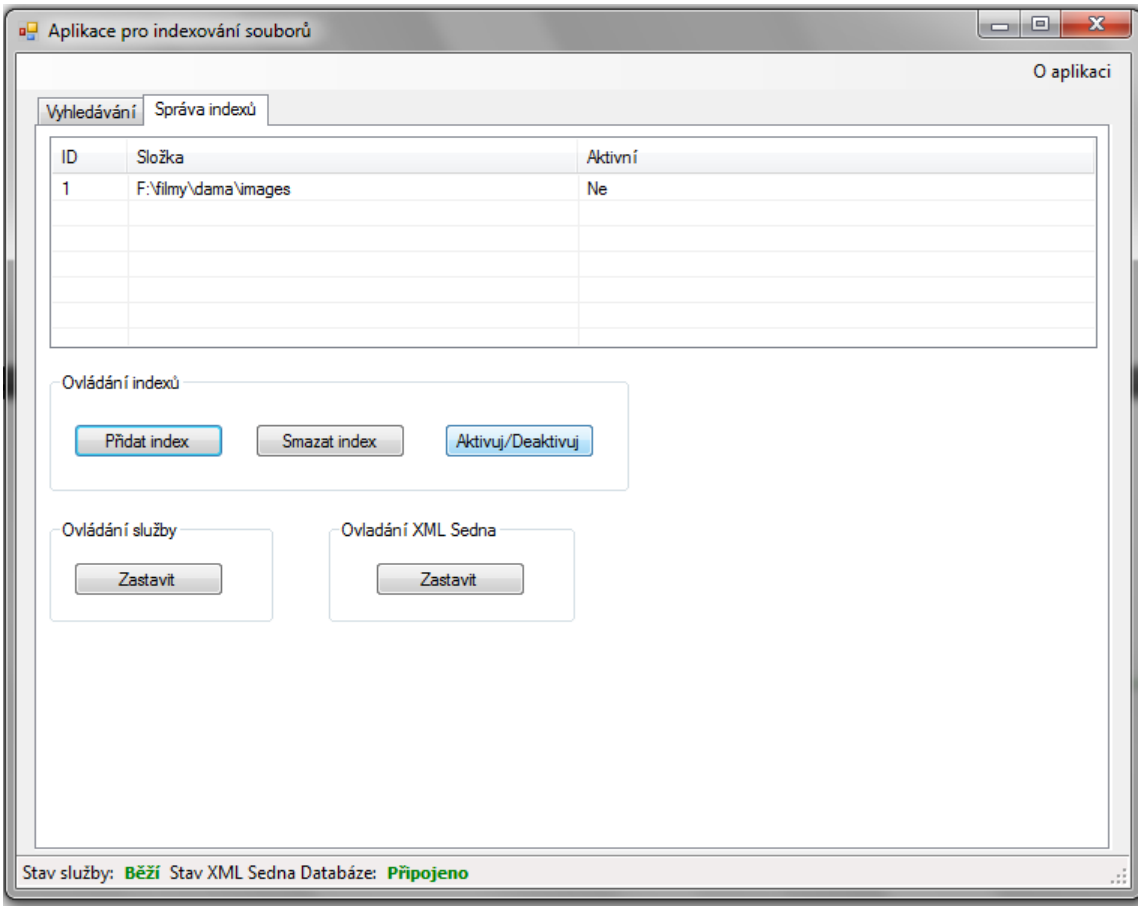
Obrázek 7: Aplikace

Ovládání služby a databáze

Ve stavovém řádku lze zjistit stav XML databáze a stav služby indexování souborů. Pomocí tlačítek ovládání služby a ovládání Sedny lze jejich běh ovládat.
Další tlačítka slouží k ovládání indexů - přidávání, odebírání a aktivaci/deaktivaci indexování.

Správa indexů

Přidání indexu provedeme stiskem tlačítka *přidat index*. Objeví se dialog, ve kterém vybereme složku, kterou chceme indexovat. Aplikace bude chvíli pracovat a pak zobrazí indexovanou složku v horní tabulce. Situaci ilustruje obrázek 8.



Obrázek 8: Aktivace indexu

Vytvořený index zatím není aktivní, to znamená, že pokud by byly v indexované složce provedené změny (např. vytvořen, smazán soubor), nebyly by tyto změny zavedeny do indexu. Aktivace se provede označením požadovaného indexu a stiskem tlačítka *Aktivuj/Deaktivuj*.

Odebrání indexu se provádí označením indexu v tabulce a stiskem tlačítka *Smazat index*.

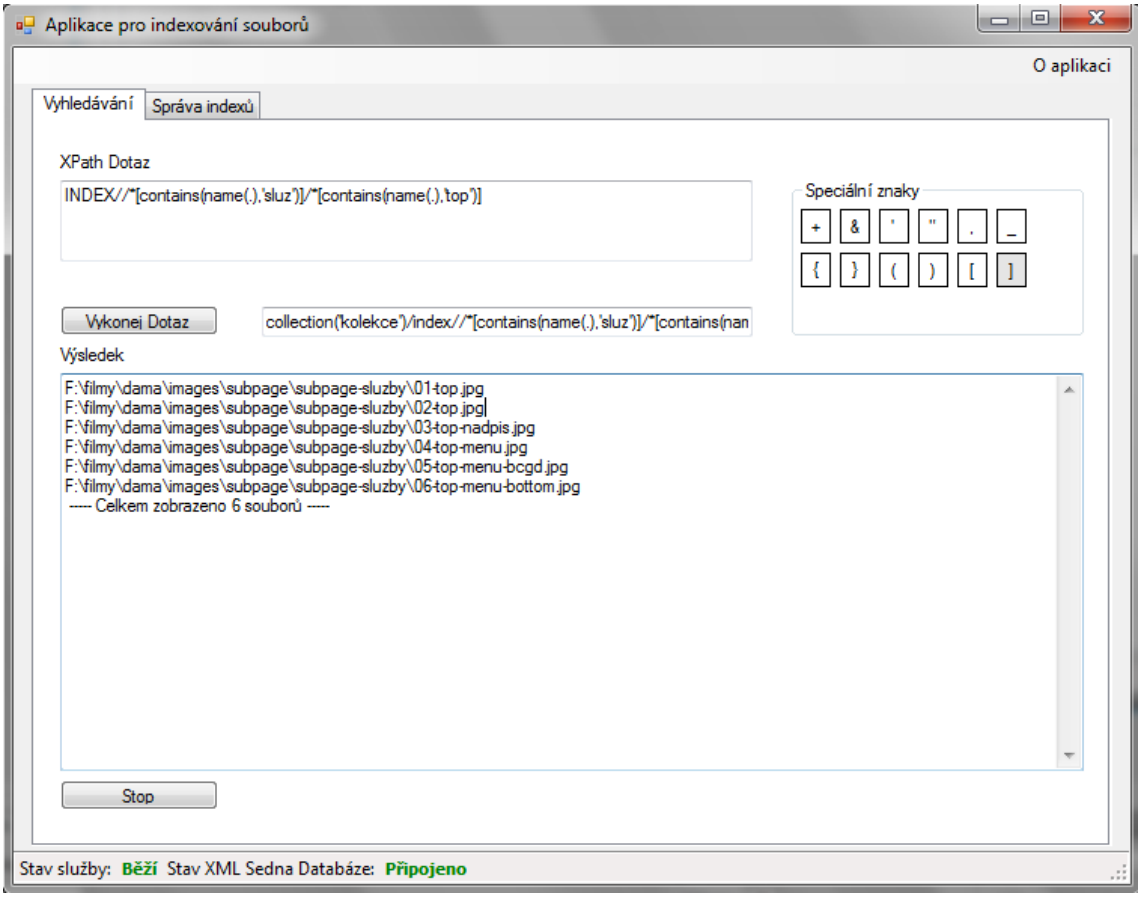
Vyhledávání v indexu

Pro vyhledávání v indexu je určena záložka *Vyhledávání*. Do pole *XPath dotaz* se zadá dotaz, který se spouští stiskem tlačítka *Vykonej Dotaz*. V poli *Výsledek* se zobrazí nalezené soubory.

Vedle tlačítka *vykonej dotaz* se nachází pole, ve kterém je částečně zobrazen upravený dotaz, který se dále ještě upraví, než je odeslán k vykonání XML databází.

V boxu speciální znaky jsou tlačítka, která slouží k vložení nahrazujících konstant za tyto znaky. Pokud má například námi hledaný soubor obsahovat v názvu znaménko „+“ musíme použít tlačítko s tímto znakem.

Tlačítko *Stop* slouží k zastavení výpisu výsledků.



Obrázek 9: Vyhledávání v indexu

Odinstalování aplikace

V nabídce programů v nabídce start lze najít zástupce s názvem Odinstalovat aplikaci pro indexování souborů. Po spuštění dojde k odregistrování služby ze systému Windows, odstranění Sedna XML databáze z paměti a odstranění dříve nainstalovaných souborů a vytvořené databáze.

C. Ukázka log souboru Sedna XML databáze

Log soubor Sedna XML databáze, která vykonává operace posílané Službou indexování souborů.

Odstranění souboru:

--- UPDATE delete doc('1','kolekce')/index/_spacer.gif

Vytvoření složky:

LOG 28/04/2010 22:55:08 (TRN Demo pid=5832 sid=47 trid=48)
[tr_functions.cpp:on_user_statement_begin:80]: User query:
--- update insert <_nova/> into doc('1','kolekce')/index

Vytvoření souboru:

LOG 28/04/2010 22:55:10 (TRN Demo pid=5832 sid=47 trid=48)
[tr_functions.cpp:on_user_statement_begin:80]: User query:
--- update insert <_eshop-odelovac.jpg path='F:\filmy\dama\images\nova\eshop-odelovac.jpg' /> into doc('1','kolekce')/index/_nova